Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER DE SPÉCIALISATION EN INFORMATIQUE ET INNOVATION

Le rôle d'un Business Analyst dans un projet de startup : Analyse et recommandations en cas de sous-traitance à l'étranger

Clause, Olivier

Award date:
2017

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

Business Analysis et Gouvernance IT

Le rôle d'un Business Analyst dans un projet de startup : Analyse et recommandations en cas de sous-traitance à l'étranger.

Auteur : Olivier Clause

Mémoire présenté en vue de l'obtention du titre de

Master de spécialisation en Informatique et Innovation

Mémoire encadré par :

Professeur Naji Habra et
Bertrand Verlaine

ANNEE ACADEMIQUE 2016-2017

Faculté d'informatique
Faculté des Sciences économiques, sociales et de gestion
Département des Sciences de gestion

Université de Namur

Remerciements

Je désire remercier le professeur Naji Habra pour le temps et la patience qu'il m'a consacré afin de réaliser ce mémoire et la connaissance de qualité qu'il m'a transmise.

Je remercie tout particuilèrement Hajer Ayed doctorante Unamur et Pawel Mysliwiec, Agile Coach and Professionnal Scrum.org Trainer pour les nombreux meetings et conférence call ainsi que l'apport de sources, de conseils avisés et de retours d'expériences qui ont été utilisés à la rédaction de ce mémoire.

Je remercie également mon employeur pour le temps qu'il m'a laissé pour mener à bien ce mémoire.

Je remercie aussi vivement Mathieu Dewulf pour le temps qu'il a consacré à la relecture du texte.

Merci également à ma femme, ma fille, ma famille et à mes amis qui m'ont soutenu moralement pendant la rédaction de ce mémoire.

Résumé

Grâce à la mondialisation et à internet, de plus en plus de projets informatiques se font avec des équipes distantes. Dans de tels projets le rôle du Business Analyst est primordial. Il est également nécessaire de comprendre le contexte et d'adopter une méthodologie adaptée. Ce mémoire propose une analyse d'un projet informatique avec une équipe offshore qui a été arrêté et explique les raisons de cet échec. Il montre également les différentes recommandations à adopter afin de recommencer ce projet et négocier bun nouveau contrat de sous-traitance avec une nouvelle équipe en adoptant des outils et des modèles adéquats.

Abstract

Due to globalization and the Internet, more and more IT projects are being carried out with offshore and/or remote teams. In this kind of project the role of the business analyst is essential. It is also necessary to understand the context and adopt a suitable methodology. This dissertation proposes an analysis of a project with an offshore team that has been stopped and explains the reasons for this failure. It also shows the various recommendations to be adopted in order to restart this project and to negotiate a new contract with a new team by adopting appropriate tools and models.

Table des matières

1. Introduction

Ce mémoire a pour but de montrer que l'absence de Business Analyst peut avoir de lourdes conséquences dans un projet informatique quel que soit sa taille.

Mon analyse se base sur mon expérience professionnelle (15 ans), l'application directe des modèles vu lors de mon cursus de MSBAGI, les littératures existantes sur le sujet et a été réalisé avec l'aide de Pawel Mysliwiec coach chez Scrum.org et Hajer Ayed doctorante à l'université de Namur.

Ce mémoire se décompose en deux parties.

La première partie du mémoire va démontrer le rôle primordial d'un Business Analyst dans chaque projet informatique même dans des petits projets de startup où cette fonction est trop souvent ignorée.

Mon analyse de départ se base sur le cas réel d'une startup dont les membres n'avaient aucunes connaissances IT et qui a donc sous-traité complètement son développement à une société externe basée à l'étranger. Après une première release complètement buggée, cette société a fait appel à mes services pour tenter de rectifier le tir. Cependant une bonne partie du budget initial ayant été complètement consommé avant mon arrivée, j'ai conseillé d'arrêter le contrat avec cette équipe. Cette première partie va donc d'abord analyser les erreurs commises et tirer les conclusions de cet échec.

Ensuite la seconde partie traite des différentes recommandations qui ont été données à l'équipe de Beauteza afin de pouvoir l'aider à négocier avec un nouveau prestataire, et de la nouvelle méthodologie de gestion de projet plus appropriée mise en place afin de à mener à bien le projet. La question se pose dès lors s'il est possible de réconcilier des méthodes de développement de type Agile qui prônent les interactions entre les individus, avec des équipes en offshore complètement distantes.

Pour cette partie je me base entre autres sur mon expérience professionnelle de supervision de développement pour de gros projets réalisés à l'étranger. J'ai en effet managé différentes équipes de développements dans de nombreux pays notamment le Maroc, L'Espagne, l'inde, la République tchèque, les Philippines, la Colombie, le Chili.

Au travers de cette capitalisation d'expériences j'ai pu analyser la culture de ces pays et me rendre compte que la façon de travailler et de s'organiser diffère totalement d'un pays à l'autre. Le problème étant que les méthodes actuelles ne tiennent quasi jamais compte de ces particularités.

Ensuite grâce à l'aide de différents modèles et de concepts vus lors du cursus du MSBAGI je vais pouvoir formaliser ces expériences et proposer une solution plus appropriée dans le cas de Beauteza.

2. Description du projet

Beauteza est une startup chilienne créée en avril 2016. Son objectif est de créer une application mobile et desktop pour le particulier permettant de trouver des salons de beauté et d'y faire une réservation en fonction des horaires disponibles.

Le taux de pénétration des smartphones au Chili étant assez important et continuant de croitre, l'objectif est de faciliter la gestion des salons via une application commune afin de se passer des traditionnels carnets de rendez-vous papier et du téléphone. Cela permet aussi d'optimiser ces carnets étant donné que les places disponibles restantes sont diffusées et peuvent faire l'objet de promotion. Pour l'utilisateur cela est intéressant aussi car il peut choisir rapidement le salon le plus proche en fonction de la disponibilité et le prix.

L'utilisateur a aussi la possibilité de faire un appel d'offre pour un service qui est envoyé à tous les salons autour de lui avec comme critère la distance.

Au niveau des revenus, les salons membres payent soit un abonnement mensuel de 10 USD soit une commission qui est prise sur chaque réservation effectuée.

Le but est de lancer cette application en premier lieu au Chili comme pays pilote et par après dans plusieurs autres pays d'Amérique latine ainsi qu'aux USA et au Canada. Au niveau de la concurrence Beauteza se démarque assez bien de la faible concurrence existante par le fait qu'elle offre des fonctions assez innovantes comme la fonction « Beauteza now ». En effet un utilisateur a la possibilité de demander une prestation de service à une heure données en envoyant un « push » à tous les salons de beauté autour de lui. Ces deniers peuvent lui répondre et lui proposer un prix. Il peut ensuite prendre rendez-vous.

Cette startup a été créé au Chili car Santiago de Chile, appelé communément « Chilecon valley » offre beaucoup d'avantages pour lancer sa Startup[1].

Grâce à un business model bien défini Beauteza a pu bénéficier d'un capital d'environ 10 000€ pour débuter ses activités via un incubateur local. Une fois l'application développée, l'entreprise recevra un montant de 50 000€ via l'incubateur Geek Camp[2] pour développer sa stratégie marketing et s'imposer sur le marché. Parallèlement à ma mission qui sera décrite par après et afin d'avoir un business plan défendable auprès de cet incubateur, j'ai également assisté Beauteza dans l'amélioration de son business model afin d'obtenir cette subvention supplémentaire.

[1] (Andersson, 2015)
[2] http://incubauc.cl/geekcamp/

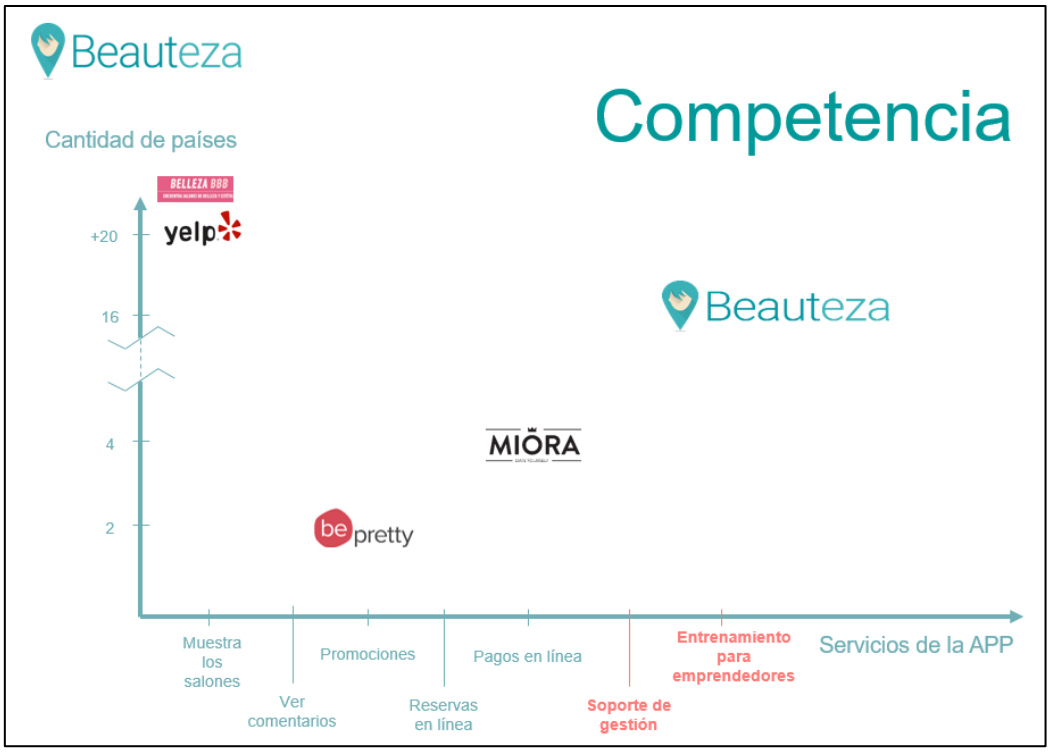
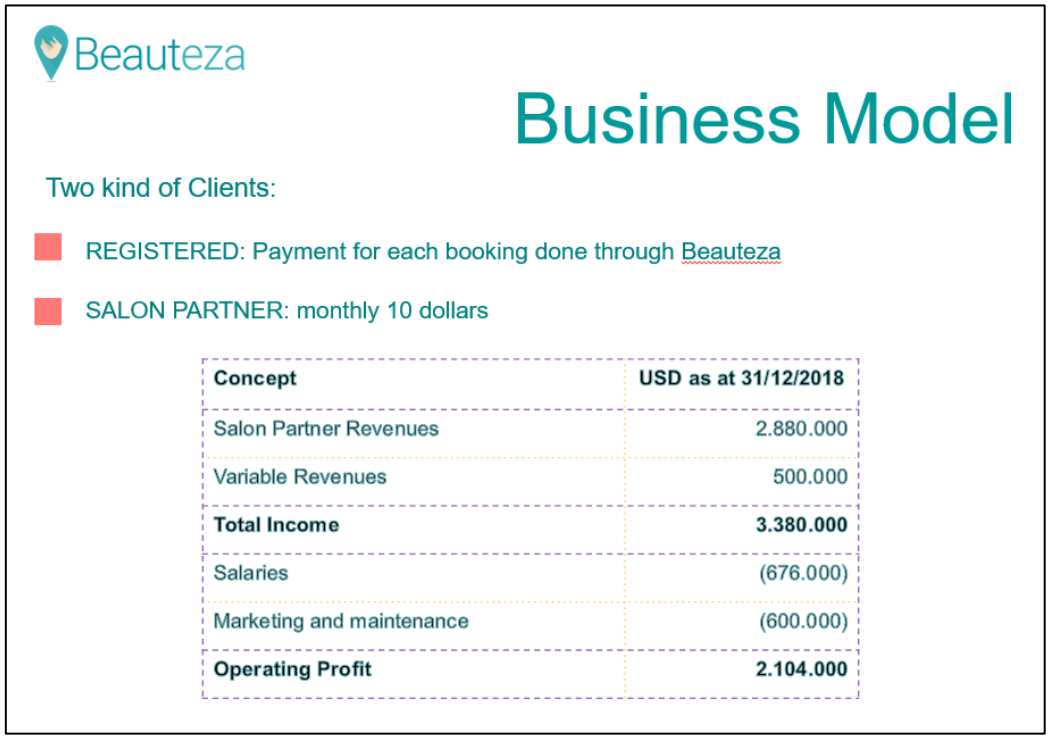


Figure 1 Concurrents Beautza

Tout d'abord il fallait situer Beauteza par rapport à ses concurrents. Dans cette figure nous pouvons voir que les services de Beauteza vont couvrir beaucoup plus de domaines que les concurrents directs ainsi qu'elle sera présente dans beaucoup plus de pays. Ce qui en fait un produit intéressant. Analysons plus en détail le business model à présent :



Concept	USD as at 31/12/2018
Salon Partner Revenues	2.880.000
Variable Revenues	500.000
Total Income	**3.380.000**
Salaries	(676.000)
Marketing and maintenance	(600.000)
Operating Profit	**2.104.000**

Figure 2 Business model Beauteza

Les sources de revenus seront de deux types. D'une part une cotisation mensuelle de 10 USD et d'autre part un pourcentage sur chaque transaction réalisée pour les salons non enregistrés.

Pour convaincre les investisseurs j'ai également suggéré d'utiliser le Business Model Canvas et la Value Proposition Canvas comme vu lors du cursus de MSBAGI[3] :

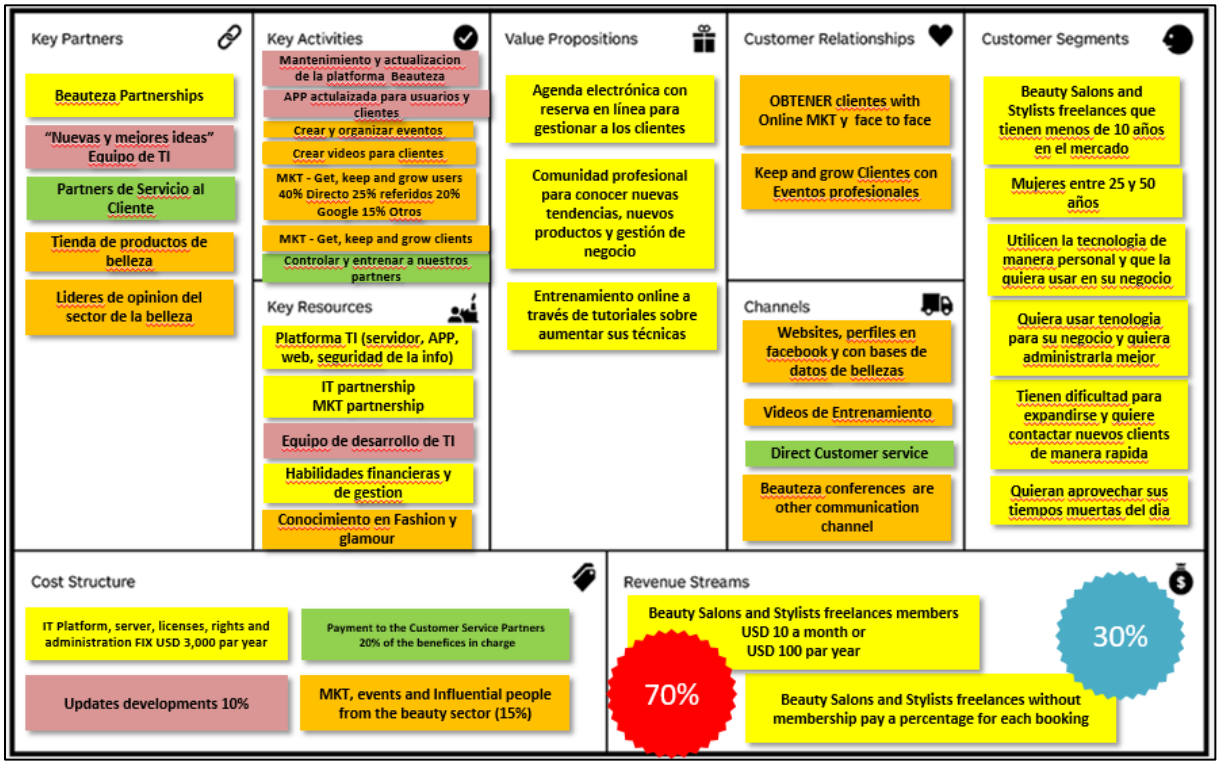


Figure 3 Business model canvas

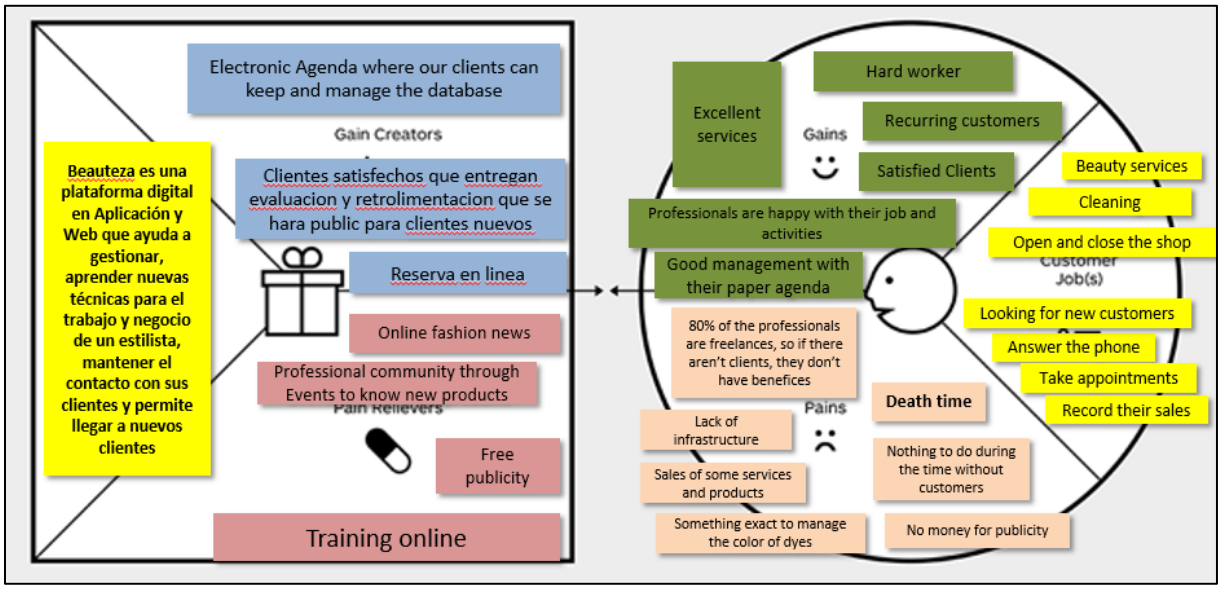


Figure 4 Value Proposition Canvas

[3] (Sieberath, 2015)

Au départ l'équipe se compose de trois personnes basées à Santiago du Chili.

Une fois le financement initial obtenu, l'équipe n'ayant aucune connaissance IT a fait un appel d'offre à l'aide d'un RFP sommaire. Une équipe de développement basée en Colombie a été choisie pour des raisons de langue et de budget.

Le coût négocié pour le développement de la solution était de 8000€ et il a été convenu avec l'équipe de développement d'avoir une livraison intermédiaire après 2 mois et une livraison finale après 4 mois. 60 % du paiement se ferait en début de projet, les 20 % suivants après la première livraison et les 20 % restants après la livraison finale.

Suite à un retard conséquent sur la première livraison, qui en plus ne correspondait pas du tout à ce qui avait été demandé et qui comportait de nombreux bugs, l'équipe Beauteza a fait également appel à mes services en tant que Business Analyst afin de tenter de « déminer » la situation.

3. Problématique et existant

Ma première activité fut d'identifier les bugs et les fonctionnalités manquantes de la première livraison. Pour ce faire j'ai demandé à voir les spécifications qui étaient en fait quasi inexistantes. Avant donc de commencer quoi que ce soit, j'ai dressé un tableau de la situation afin de mettre en évidence les problèmes majeurs. Voici donc les éléments que j'ai relevé :

3.1. Problème de communication et de compréhension entre l'équipe de développement et le business

Les échanges se faisaient via différents canaux de communication (mail, messagerie WhatsApp) et n'étaient pas structurés. Il était très difficile de retrouver un historique de toutes les discussions.

3.2. Cahier des charges et analyse faible

L'analyse et les spécifications se résumaient uniquement à une maquette et les tâches étaient définies via la plateforme Asana[4]. Le planning consistait uniquement en un document word et il n'y avait aucune stratégie de tests clairement définie

3.3. Rôle de Business Analyst inexistant

Le problème majeur fut que les deux parties ne savaient pas bien dialoguer entre elles. L'équipe développement avait un vocabulaire technique incompréhensible pour l'équipe de Beauteza et inversement les « requirements » fonctionnels n'étaient pas bien compris par l'équipe en Colombie. Un rôle de Business Analyst devenait alors indispensable. Une situation conflictuelle était née il fallait également quelqu'un capable de la résoudre.

Bien que cette fonction de Business Analyst est cruciale elle n'est pas encore assez répandue dans les pays industrialisés et elle est même carrément absente dans les pays émergeants comme le Chili.

Pour rappel le rôle du Business Analyst tel que défini au cours de négociation de conflit[5] doit être le suivant :

Considérations générales

- Appartient généralement au *business*
- Il est le lien privilégié entre l'IT et les autres personnes du projet
- Il établit la liste des exigences (formuler des hypothèses)
- Il coordonne les équipes (si aussi responsabilités de chef de projet)
- Il s'occupe de faire valider le client
- Compétence IT (pas obligatoires)

[4] http://asana.com/
[5] (Maquet, 2015)

Compétences orientées « Ingénierie des exigences » et analyse de manière plus générale

- Il doit être un bon communiquant (bi ou trilingue)
- Au niveau Interview il doit savoir que le maximum d'information se trouve dans les 5, 10 premières phrases
  - Il s'agit de l'expression du problème mais peut-être pas du besoin
  - Les données doivent être objectives
- Il doit faire parler le client
- Il doit savoir reformuler pour être sûr qu'il a bien compris
- Importance d'écouter ce que les autres disent
  - On peut profiter des préoccupations des autres
  - Idées nouvelles donc plus de créativité

Compétences orientées « Négociation »

- Empathie
- Le sang-froid
- L'art du calcul
- Le sens du recul – la prise de recul
- La pratique de l'écoute active
- La maîtrise de la stratégie
- L'utilisation des bonnes techniques
- Un bon Business Analyst doit avoir une longueur d'avance

4. Tentative de résolution

Etant donné qu'il était impossible de mener à bien ma mission avec ce constat, j'ai proposé aux deux équipes de faire une pause afin de clarifier la situation et apporter une solution convenant aux deux parties. Ma solution était la suivante :

4.1. Mise en place d'outils de communication efficaces

Les différents moyens de communication assez hétéroclites ont a amené beaucoup de confusion, c'est pourquoi il a fallu mettre en place un moyen de communication standardisé. Pour ce faire le logiciel slack [6] a été proposé et accepté par les deux équipes. Grâce à celui-ci on avait maintenant un moyen de communication central et efficace avec lequel toutes les discussions étaient archivées en un seul endroit.

4.2. Analyse des exigences

Les spécifications étant absentes il a été nécessaire de les écrire complètement.

Afin de mener une analyse des exigences efficace, les différents procédés et outils enseigné lors du cours d'ingénierie des exigences ont été utilisés [7]:

4.2.1. Collectes des exigences

Afin de comprendre le domaine, et collecter l'information les 2 méthodes suivantes ont été utilisées par le Business Analyst :

- Interviews non structurées avec l'équipe de beauteza
- Workshop avec les 2 équipes

4.2.2. Spécifications

Une fois l'élicitation des exigences terminée, j'ai utilisé les outils UML suivants afin de dégager un modèle conceptuel :

- Diagramme d'activité
- Use Cases
- Class diagram

[6] https://slack.com/
[7] (Burnay, 2016)

Ces documents ont alors fait l'objet d'un nouveau workshop avec l'équipe Beauteza et ont été validés par celle-ci. Une fois cette étape accomplie, ces documents (cfr Annexes) ont été transmis à l'équipe développement et un dernier workshop a été organisé afin d'avoir une validation définitive de celle-ci.

Sur base de cette analyse un planning réaliste a été défini et approuvé par les deux parties.

4.3. Négociation et conflits avec l'équipe de développement

Suite à ma venue dans le projet, mon rôle externe de Business Analyst imposé à l'équipe de développement a apporté une méfiance de celle-ci. La première étape a donc consisté à résoudre ce conflit potentiel et à gommer les incertitudes quant à ma fonction dans le projet. De plus il était dans l'intérêt de tous d'obtenir la pleine collaboration de l'équipe de développement et il a donc fallu en premier lieu identifier les peurs et craintes potentielles et entamer une négociation afin d'arriver à un objectif commun. [8] Je voulais absolument éviter de me retrouver dans le schéma du triangle dramatique avec le rôle de sauveur-persécuteur-victime[9].

Pour identifier le problème et arriver à un accord juste, j'ai utilisé l'outil RPBDNC tel que vu dans le cadre du cours de négociation et analyses des conflits [10] :

Les étapes de ce processus étaient les suivants :

- Réel : La tension entre l'équipe de Beauteza et l'équipe de développement est papable.
- Problème : La première version de l'application a été livrée en retard et ne correspond pas à ce qui a été demandé.
- Besoin : l'équipe de beauteza a besoin d'avoir l'application rapidement opérationnelle afin de disposer des fonds supplémentaires pour lancer sa stratégie marketing. L'équipe de développement quant à elle avait besoin de la validation de la première livraison afin d'être rémunéré pour le travail accompli
- Demande : la demande de Beauteza était claire. Il était impératif d'avoir une application fonctionnelle dans 4 mois.
- Négociation : La négociation a duré 2 réunions. Il a été convenu que Beauteza n'arrêtait pas le contrat, mais en contrepartie l'équipe de développement a accepté un arrêt du projet de 1 semaine afin de permettre la définition claire des exigences dans un langage compréhensible par tous. Ces exigences ont été écrites en UML.
- Contrat : Une fois les exigences écrites, comprises et validées, l'équipe de développement a fourni un nouveau planning qui a été validé par les deux parties. Sur base de cela un accord a été conclu.

[8] (Maquet, 2015)
[9] (Karpman, 1968)
[10] (Maquet, 2015)

L'accord négocié était le suivant :

- Une analyse des exigences a été faite via un activity diagram, des use cases et un class diagram. La maquette existante servira à vérifier les enchainements et le design.
- Sur base de cette analyse, un nouveau planning a été définie et validé.
- Une release devra être délivrée toute les 2 semaines et couvrira des uses cases définis lors de l'analyse. Celle-ci devra être accompagnée d'une note explicative (release note). Cette release devra être déployée sur Android, IOS et via une page web.
- Lors de chaque release, une vidéo conférence sera établie afin de faire une démonstration de celle-ci et montrer qu'elle couvre bien les scenarios demandés.
- Les tests vérifieront les use cases et un format de document commun aux 2 parties devra être établi.
- L'équipe de Beauteza aura 3 jours pour réaliser les tests et approuver la release. Si celle-ci n'est pas validée, elle devra fournir le document avec les tests réalisés.
- Si la release n'est pas validée, l'équipe de développement devra corriger les erreurs et fournir une version corrigée lors de la release suivante.

Cette négociation a été validée et vérifie les 5 critères de réussite suivants tels que vu dans le cadre du cours de négociation et analyses des conflits [11] :

1. Le problème est réglé en profondeur et pas superficiellement ?

Oui. Les rôles, droits et devoirs de chacun ont été clarifiés ce qui a permis de conduire une nouvelle analyse fonctionnelle et donc de continuer le projet. Les processus d'analyse ont été clairement établis et l'objectif fixé est SMART (Spécifique, Mesurable, Acceptable, Réaliste et Temporel).

2. La solution est réellement applicable sur le terrain par les acteurs concernés ?

Oui. Les acteurs ont désormais un processus clair ainsi qu'un objectif précis. Le Business Analyst a mené à bien la seconde analyse et le client s'est engagé à y participer activement.

3. La solution est équitable ?

Oui. La solution parait équitable. Malgré le fait que les coûts sont à la charge de l'équipe de développement, il était nécessaire de faire cette analyse afin d'éviter un échec.

4. L'accord est durable ?

Oui. L'accord été respecté et la seconde analyse a été terminée et validée par les deux parties.

5. Il y a satisfaction des deux parties qui ont chacune l'impression d'avoir gagné quelque chose.

Oui, du côté de l'équipe Beauteza, il s'est avéré clair qu'une analyse approfondie des exigences était indispensable afin de bien exprimer le besoin et du côté de l'équipe de développement, cette analyse était également nécessaire afin d'établir un planning réaliste et fournir et couvrir toutes les fonctionnalités demandées par le client.

[11] (Maquet, 2015)

4.4. Définition d'une stratégie de test

Une stratégie a été définie pour les tests sur base de scénarios découlant des use cases.

Les tests se sont déroulés en 3 parties. Ils étaient composés de tests unitaires, de test de non régression et enfin de test d'intégrations.

4.4.1. Tests unitaires

Chaque test unitaire devait vérifier un use case. Ces tests devaient être exécutés avant chaque livraison et devaient suivre un template bien précis défini à l'avance lors de la phase d'analyse. Ces tests devaient être livrés en même temps que la release.

4.4.2. Tests de non régression

Lors de chaque nouvelle release, un script de test était exécuté afin de vérifier qu'ils fonctionnent toujours.

L'équipe de développement était également en possession de ces scripts de façon à évaluer leur développement de manière continue.

4.4.3. Test d'intégration

Ces tests étaient exécutés par l'équipe Beauteza et l'ensemble de ces scénarios était compilé dans un fichier Excel.

Ces scenarios de test étaient communiqués à l'avance à l'équipe de développement et étaient exécutés après chaque livraison de release par l'équipe de Beauteza.

Les résultats de ces tests étaient transmis quelques jours après à l'équipe de développement pour correction lors de la prochaine release.

4.4.4. Tests d'environnement

Pour ce faire, une fois l'application en production, une chaine de salons de beauté avait été sélectionnée pour être pilote pour l'application pendant quelques mois.

5. Echec du projet

Malgré un accord défini précédemment, après trois releases le résultat n'était pas au rendez-vous (90 % des tests en échec), et il y avait également un non-respect des accords convenus (annulation des réunions, non respects des échéances, non-exécution des tests unitaires et aucun feedback sur les tests d'intégration).

De plus le turnover de l'équipe de développement était assez important, plusieurs développeurs ayant été remplacés, ce qui a également allongé les délais.

Ensuite le prestataire a demandé un budget supplémentaire vu l'ajout de spécifications et le temps supplémentaire demandé pour ce projet.

Suite à cela un audit de code a été demandé et a été réalisé, non sans mal car il a fallu batailler ferme pour obtenir les sources vu l'absence de ce point dans le contrat.

Après analyse de ce code source, la découverte de beaucoup d'erreurs a confirmé le manque d'expérience et de professionnalisme de l'équipe (non-respect des règles du Framework, gestion des erreurs absentes, non découpage en méthode, hardcodage de mot de passe, …).

Etant donné que ce projet était complètement sous-traité (développement et maintenance), le Business Analyst a suggéré à Beauteza de rompre le contrat car l'équipe développement avait prouvé qu'elle n'était pas un partenaire de confiance. Le contrat négocié contenant une clause spécifique de remboursement des montants libérés en cas d'échec, Beauteza va être remboursé et une nouvelle équipe de développement doit être trouvée impérativement.

D'autre part j'ai remarqué également que les besoins business au fil des différentes conférences et meeting n'était pas figés et étaient en constante évolution. De plus pour obtenir le financement du geek camp[12] pour le marketing il était nécessaire d'avoir une application fonctionnelle le plus rapidement possible. La gestion de projet de type waterfall ne s'appliquait pas dans ce cas-là.

[12] http://incubauc.cl/geekcamp/

6. Nouvelle stratégie et rôle du Business Analyst

6.1. Redéfinition des objectifs.

Il est important que le Business Analyst intervienne non seulement dans l'analyse mais aussi à différentes autres étapes du projet en tant que conseiller.

6.1.1. Minimum Viable Product

Le produit minimum viable (anglais : Minimum Viable Product, MVP) est une stratégie de développement de produit, utilisée pour de rapides et quantitatifs tests de mise sur le marché d'un produit ou d'une fonctionnalité[13]

Dans le cas de Beauteza, étant donné que la majorité du budget a été consommé avec la première équipe de développement, et ne pouvant pas attendre de récupérer l'argent auprès de ces derniers avant de commencer avec une nouvelle équipe, il est nécessaire de se pencher sur les fonctionnalités indispensables pour une utilisation commerciale. Pour se faire une classification des exigences doit être faite.

Il est donc nécessaire de dégager une valeur d'affaires le plus rapidement possible. Et pour se faire il faut donc un produit minimum viable en production, dès que possible.

6.1.2. Redéfinition du cahier des charges et de l'analyse des exigences

Afin donc d'avoir un MVP rapidement, il est primordial de redéfinir les exigences et de les prioriser afin de se focaliser en premier sur les plus importantes. Pour ce faire l'équipe de Beauteza a mené une enquête auprès de divers salons de Beauté pour savoir ce qu'une application pouvait leur apporter en premier lieu. Il en est ressorti qu'ils recherchaient en premier lieu un moyen de faire venir la clientèle pour remplir les plages vides de leur agenda. Sur base de cette enquête il a fallu trier et prioriser les exigences. Il existe une multitude d'outils mais celui qui correspond au mieux au cas présent est la matrice MoSCoW[14].

6.1.2.1. MoSCoW

La méthode MoSCoW est une technique visant à prioriser des besoins ou des exigences en matière d'assistance à maîtrise d'ouvrage et de développement logiciel. L'objectif est que le maître d'œuvre et le maître d'ouvrage s'accordent sur l'importance des tâches à réaliser par rapport aux délais prévus[15].

[13] (Thiran, 2017)
[14] (Burnay, 2016)
[15] (Burnay, 2016)

Les lettres majuscules de l'acronyme MoSCoW signifient :

- M : must have this, c'est-à-dire 'doit être fait' (minimum vital).
- S : should have this if at all possible, c'est-à-dire devrait être fait dans la mesure du possible (essentiel).
- C : could have this if it does not affect anything else, pourrait être fait dans la mesure où cela n'a pas d'impact sur les autres tâches (confort).
- W : won't have this time but would like in the future, ne sera pas fait cette fois mais sera fait plus tard' (luxe, c'est votre zone d'optimisation budgétaire).

En Annexe, se trouve la matrice MoSCoW réalisée pour Beauteza afin de déterminer le scope du Mininum viable product.

Sur base de cette analyse il a été défini que le MVP de l'application sera :

Coté utilisateur :

- Une application IOS et Android disponible en espagnol et en anglais
- La création, suppression, modification de compte
- La fonctionnalité Beauteza Now, c'est-à-dire lancer une alerte aux salons enregistrés situés dans un périmètre donné. Cette alerte comportera le type de service demandé et la tranche horaire disponible de disponibilité. Une fois que les salons reçoivent cette alerte, ils peuvent répondre en proposant un tarif et une tranche horaire. L'utilisateur reçoit ensuite toutes les possibilités dans un délai donné avec toutes les informations nécessaires du salon et confirme une proposition.
- Accessoirement la possibilité de situer les salons de beauté ayant répondu sur une carte.

Côté salon :

- Un portail Web en espagnol et en anglais
- La création, modification et suppression de compte
- La consultation des alertes Beauteza Now.

La source de revenu quant à elle sera pour le moment la cotisation mensuelle de 10 USD.

6.2. Choix d'une méthodologie efficace de gestion de projet et expérience de l'équipe dans celle-ci.

La méthode de gestion de projet employée précédemment était une méthode de type waterfall.

Dans un projet waterfall le cycle de développement suit une liste d'étapes avec l'ordre suivant[16] :

1. Analyse des besoins avec le client
2. Définitions des spécifications des fonctionnalités
3. Établissement d'un budget et d'un planning
4. Implémentation
5. Tests et corrections

[16] (Kolp, 2016)

6. Maintenance

Dans ce modèle, le processus recommence son cycle entre les étapes 4 à 6, jusqu'à ce que le logiciel soit livré au client. Malheureusement cette méthodologie est trop rigide et ne s'adapte pas bien au développement d'un application mobile étant donné par exemple que les spécifications peuvent être amenées à changer tout au long du développement.

En effet, pour être complet, on constate qu'avec ce genre de méthode deux types de problèmes surviennent assez régulièrement :

- Les besoins du client qui évoluent

L'analyse des besoins se fait au début du projet, sur une période qui s'étend sur plusieurs semaines. Ces requirements sont ensuite développés, testés et livrés. Cependant plusieurs mois se sont écoulés entre la livraison et l'analyse et les besoins peuvent avoir changé entretemps. Avec la méthodologie waterfall il peut donc y avoir un gap entre ces deux périodes et la solution fraichement développée ne peut parfois plus correspondre aux besoins du client au moment du client. Cela entraîne donc de longues et couteuses procédures de change request.

- Manque de flexibilité

Dans la méthodologie waterfall, étant donné que l'analyse des besoins est réalisée à l'avance, il n'est pas possible lors de l'implémentation de revenir en arrière et de changer un requirement en cours de route sans établir également un change request qui peut être couteux en temps (nouvelle analyse) et en argent. En effet tout nouveau changement pourrait avoir un impact sur d'autres fonctionnalités déjà développées ou encore à développer, qu'il faudrait ré-analyser. Le modèle de Waterfall n'est donc pas propice au changement. De plus il n'est pas possible d'avoir non plus une partie de l'application en production et utilisable avant la fin de son développement.

Un changement ou une anomalie survenant à ce moment-là augmente le risque de façon assez importante puisque ceux-ci sont détectés tardivement, et donc le retour arrière sera plus complexe, sa correction coûtera cher et les effets de bords seront menaçants. [17]

Dans le cas de Beauteza les exigences définies il y a 8 mois ne sont peut-être plus d'actualité maintenant et peuvent encore changer. De plus la nécessité d'obtenir rapidement un produit minimum viable pour débloquer la seconde aide financière de l'incubateur est également primordiale.

Comme nous allons le voir par la suite, pour cette problématique une méthode de projet de type « Agile » s'avère donc beaucoup plus adéquate.

6.3. Coûts indirects

Outre les coûts classiques (serveurs, nom de domaines, frais de publication sur l'AppStore...) il faut savoir que, plutôt que réinventer la roue, la majorité des applications développées aujourd'hui font un recours intensif aux API[18].

[17] (Messager Rota, 2008)
[18] (Thiran, 2017)

En effet l'essor du Cloud entraine de plus en plus le développement des services connectés et ceux-ci ont recours aux API pour faciliter l'exploitation du flux de données et accélère le développement.

Le coût de ces API est donc à prendre en considération dans le budget comme coût mensuel.

Dans le cas de Beauteza, le développement va nécessiter de faire appel aux API Google. L'utilisation de ces API est gratuite jusqu'à un certain seuil d'utilisation et est ensuite facturée[19]. Des KPI doivent être définis et ceux-ci doivent être monitorés afin de ne pas avoir de soucis d'indisponibilités lorsque l'application sera en production

6.4. Choix d'un partenaire de confiance

De cet échec nous pouvons retenir que le choix d'un prestataire pour réaliser un développement est loin d'être évident. Il ne s'agit pas simplement de choisir celui qui offre le tarif le plus compétitif mais retenir celui avec qui une collaboration à long terme peut être possible. Sans le cas de Beauteza, le scope ne s'arrête pas seulement au développement, mais il y a également une partie de maintenance à envisager. Et pour qu'elle soit efficace, elle doit se faire idéalement avec la même équipe. Cette équipe doit avoir également une expérience avec la technologie utilisée.

Lorsque plusieurs alternatives se présente une analyse SWOT peut être réalisée en prenant compte les différents éléments vus précédemment ainsi que les solutions apportées par chaque équipe (choix de la technologie utilisée, etc…)

6.4.1. Proof of concept

En fonction de la taille du projet un proof of concept peut être mis en place afin de vérifier les compétences et la capacité de l'équipe de développement à livrer une solution fiable. Le résultat de celui-ci confortera le choix du partenaire de confiance.

6.4.2. Aide à la réalisation du contrat

N'étant pas juriste, le rôle du Business Analyst dans cette partie est plutôt un rôle de conseil. La formation du contrat est une étape obligatoire et il est important qu'il soit bien réalisé avec le partenaire afin d'éviter les erreurs rencontrées dans le cas de Beauteza. De plus il s'agit d'un contrat informatique qui diffère d'un contrat classique. Indépendamment du droit propre à chaque pays il est important que le contrat mentionne bien les clauses de délivrance, de garanties et de fin de contrat comme expliqué dans le cours de droit des TIC[20].

[19] https://developers.google.com/maps/pricing-and-plans/?hl=Fr
[20] (Van Enis, 2017)

6.4.2.1. Escrow

Un dépôt fiduciaire (en anglais escrow payment) est un arrangement conclu en vertu de dispositions contractuelles entre les diverses parties d'une transaction et pour lequel un tiers de confiance indépendant reçoit et débourse l'argent ou les documents pour les diverses parties prenant part à la transaction[21].

Cette étape est indispensable pour éviter les problèmes rencontrés avec la première équipe de développement dans le cas de beauteza. En cours de contrat et en cas de cessation d'activité du prestataire, le travail peut-être poursuivit avec un nouveau fournisseur en reprenant le code source existant. Il s'agit également de pouvoir accéder à ce code en cas d'audit. Enfin il s'agit d'assurer également la bonne délivrance du code source en fin de contrat.

6.4.2.2. Le contrat de maintenance

Le contrat de maintenance doit être rédigé indépendamment du contrat pour le développement. Celui-ci doit contenir au minimum le début et la durée du contrat. Il est important également de s'accorder sur les SLA afin de définir les modalités et les délais d'intervention. Il doit y figurer aussi les modalités de paiement et les modalités d'exécution de contrat.

[21] (Wautelet, 2016)

7. Réalisation d'un projet Agile avec une équipe de développement sous-traitée à l'étranger.

7.1. Justification du choix de la méthode Agile

Qu'est-ce qu'une méthode Agile ?

« Une méthode Agile est une approche itérative et incrémentale, qui est menée dans un esprit collaboratif, avec juste ce qu'il faut de formalisme. Elle génère un produit de haute qualité tout en prenant en compte l'évolution des besoins des clients »[22].

Les méthodes Agiles sont apparues en 2001 aux Etats-Unis. Dix-sept experts en développement logiciel se sont réunis afin de mettre au point ces méthodes suite à un taux d'échec important des projets informatiques dans les années 90.

Ce rassemblement a donné naissance à un Manifeste définissant quatre valeurs[23] :

- Les individus et leurs interactions avant les processus et les outils
- Des fonctionnalités opérationnelles avant la documentation
- Collaboration avec le client plutôt que contractualisation des relations
- Acceptation du changement plutôt que conformité aux plans

Comme défini lors du cours de gestion de projet et de gestion de risques[24], les méthodes Agiles utilisent un principe de développement dit itératif qui consiste à découper le projet en plusieurs étapes qu'on appelle communément « itérations ».

Ces itérations sont en fait des sous-projets définis avec le client en détaillant les différentes fonctionnalités qui seront développées en fonction de leur priorité. En effet il est assez difficile, voire impossible de tout prévoir en début de projet et la réalisation du projet en itération plutôt qu'en une seule traite permet d'incorporer des changements en cours de route.

Les avantages du développement itératif sont donc multiples[25] :

- Meilleure qualité de la communication : L'utilisateur à la possibilité de clarifier ses exigences au fur et à mesure du développement.
- Meilleure visibilité : Le client a une meilleure visibilité sur l'avancement des travaux.
- Meilleur contrôle de la qualité : Les tests sont effectués en continu ce qui améliore grandement la qualité.
- Meilleure détection des risques : Les risques sont généralement détectés plus tôt.
- Contrôle des coûts : Le projet peut être arrêté s'il n'y a plus de budget tout en ayant déjà des fonctionnalités développées et utilisables.
- Motivation et confiance de l'équipe : Satisfaction d'atteindre un objectif fixé.

[22] (Messager Rota, 2008)
[23] (Jeff Sutherland K. S., 2001)
[24] (Kolp, 2016)
[25] (KAMDEM, 2016)

7.2. Facteurs de sélection de la méthode Agile

Il existe également des facteurs du contexte qui impactent les projets et qui ne sont pas assez pris en compte par les cycles de type waterfall.

Dans leur ouvrage Boehm et Turner définissent un modèle de cinq facteurs critiques qui justifient l'utilisation des méthode Agile [26] .

Ce modèle identifie les risques relatifs au projet et indique s'il est préférable d'utiliser une méthode Agile ou plutôt une méthode traditionnelle et structurée de type waterfall, par exemple.

Le modèle consiste en cinq étapes :

Analyse du risque, Comparaison du risque, Analyse de l'architecture, Ajustement du cycle de Vie et Exécution et monitoring. Pour ce faire Boehm et Turner ont identifié cinq facteurs d'agilité représentés par des axes d'analyse qui interviennent dans la sélection du type de méthode :

- la taille c'est-à-dire le nombre des membres de l'équipe,
- la criticité du projet,
- le dynamisme qui traduit le degré de changement des spécifications,
- le personnel ou degré de compétence et d'expérience de l'équipe,
- la culture de l'organisation.

Chaque axe représente une dimension. Quand les données du projet sont évaluées sur les axes, les différents points sont reliés entre-deux et les formes obtenues sont analysées. Si la forme est centrée, le modèle suggère l'utilisation d'une méthode Agile. Une forme qui tend vers les extrémités indique l'usage d'une méthode classique avec un planning défini.

Ce modèle peut être utilisé comme un mécanisme pour commencer l'évaluation de la faisabilité d'Agile. Cependant, il ne présente pas les détails quant à l'application des pratiques Agiles. Le défi majeur réside dans l'adaptation des pratiques Agiles à leur contexte de développement et dans l'application des pratiques les plus appropriées dans le cadre des activités de l'entreprise.

[26] (Boehm & Turner, 2003)

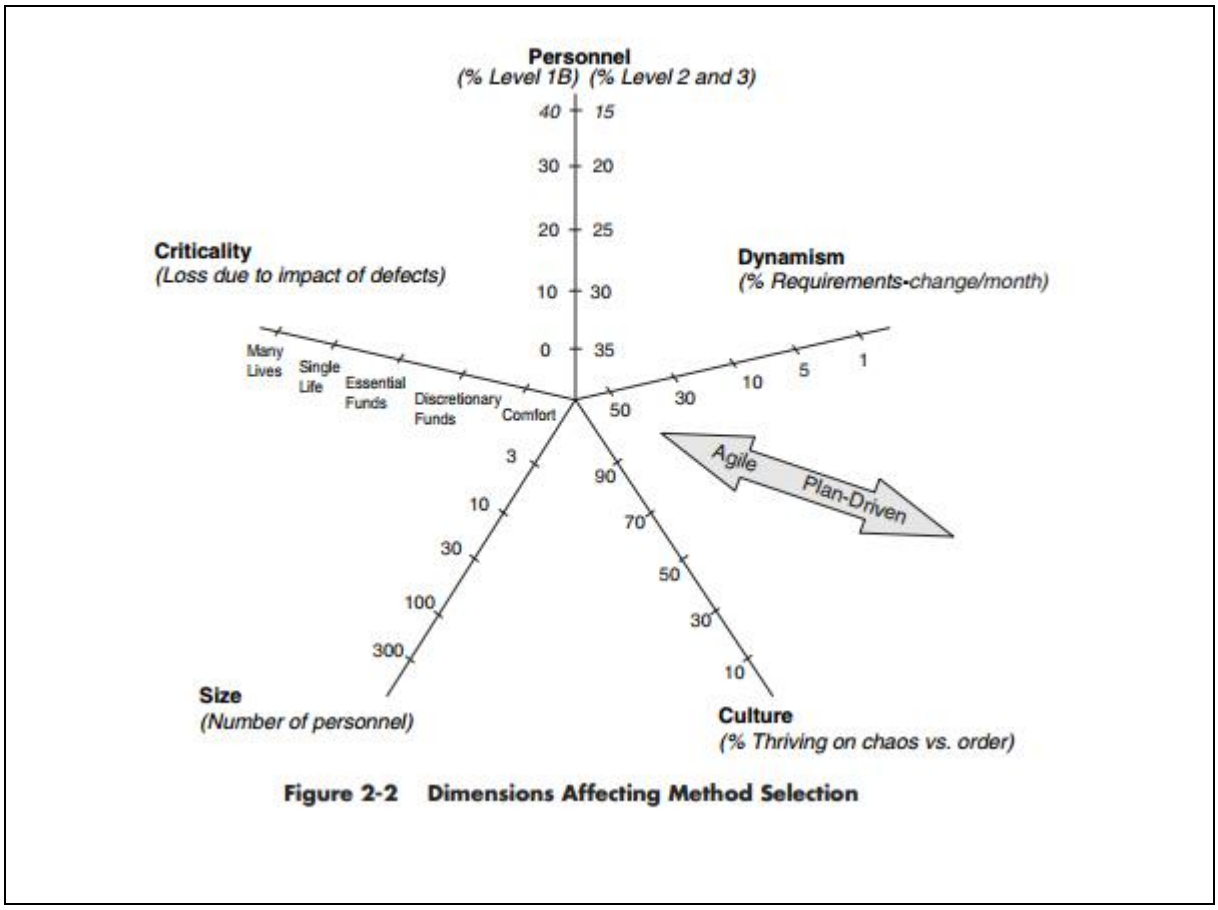


Figure 5 - Les 5 facteurs d'agilité de Boehm & Turner[27].

Dans le cas de Beauteza, on obtient une forme qui est orientée vers le centre et donc l'emploi d'une méthodologie Agile se justifie pleinement.

7.3. Le contexte

Les méthodes Agiles dépendent du contexte et doivent s'adapter à celui-ci. Il est bien évident que les pratiques Agiles mises en place ne peuvent pas être les mêmes et qu'elles doivent tenir compte du contexte. Non seulement la liste des pratiques n'est pas reproductible d'un projet à un autre mais l'ordre dans lequel elles sont introduites varie aussi.

Selon Philippe Kruchten[28] tous les aspects des projets sont affectés par le contexte : la taille, la distribution d'équipe, la criticité, l'existence d'une architecture, la gouvernance, le business model et le taux de changement vont guider les bonnes pratiques utilisées. Sur base de cette constatation le modèle octopus a été défini :

[27] (Boehm & Turner, 2003)
[28] (Kruchten, Contextualizing Agile Software Development, 2010)

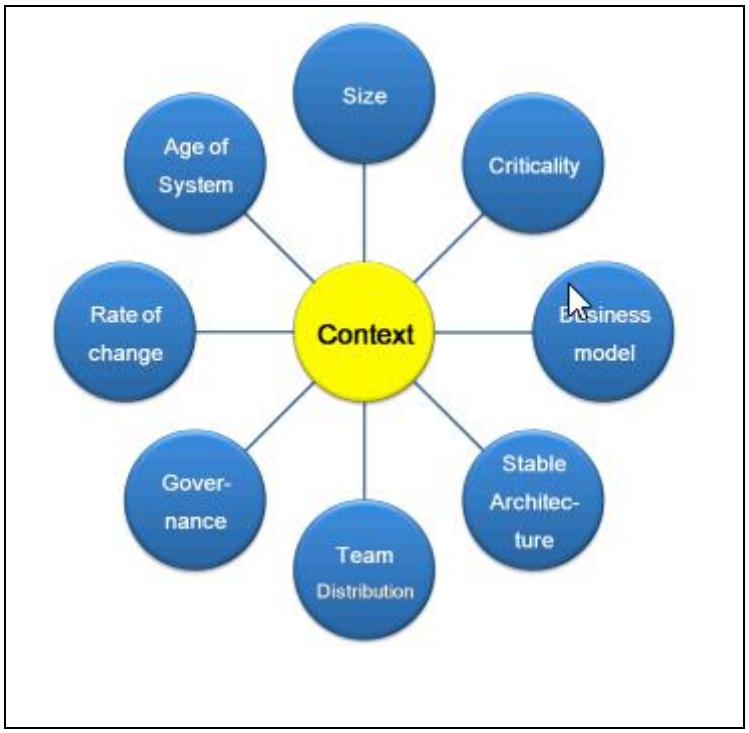


Figure 6 - Le modèle octopus[29]

Dans le cas de Beauteza le contexte peut être le taux de changement ou le business model mais surtout la distribution de l'équipe. Il est donc important d'adapter une methode Agile sur base de ce contexte.

Il est important de souligner qu'une thèse de doctorat à l'Université de Namur est en train d'être réalisée par Hajer Ayed afin de dégager une matrice de modèles sur base des contextes et quelques publications ont été faites à ce propos [30] [31] [32].

7.4. Avantages de la sous-traitance à l'étranger

7.4.1. Faible coût de la main d'œuvre

Le but premier de la sous-traitance est de réduire les coûts de réalisation d'un projet informatique. La mondialisation et les technologies de la communication et de l'information nous ont donné ces dernières décennies la possibilité de pouvoir travailler avec une main d'œuvre étrangère très bon marché. De nombreuses entreprises ont complétement délocalisé leur centre de développement dans des pays moins industrialisés où les salaires sont moins élevés, comme l'Inde ou le Maghreb.

[29] (Kruchten, Contextualizing Agile Software Development, 2010)
[30] (Ayed, 2012)
[31] (Ayed, Habra, & Vanderose, AM-QuICk : a measurement-based framework for agile methods customisation, 2013)
[32] (Ayed, Habra, & Vanderose, A Context-Driven Approach for Guiding Agile Adoption: The AMQuICk Framework, 2015)

7.4.2. Recherche de compétences locales (à l'application)

En Belgique la majeure partie des entreprises informatiques sur le territoire sont des entreprises de consultance qui fournissent du conseil ou implémentent des logiciels déjà développés (SAP,…). Il n'existe quasi plus de société qui font du développement à proprement dit[33].

On est donc parfois contraint de sous-traiter à l'étranger car il est impossible de trouver la main d'œuvre nécessaire pour une technologie particulière au niveau du marché local.

7.5. Problèmes récurrents de la sous-traitance à l'étranger.

7.5.1. Distance

La distance ne se résume pas seulement à une séparation physique, elle couvre également un manque de contact, un décalage horaire, un problème de langue ainsi que la communication. Tous ces points doivent être pris en compte dans un projet informatique sous-traité à l'étranger.

7.5.2. Turnover

Le turnover peut être assez important. En effet dans certains pays, un seul bâtiment abrite plusieurs compagnies et c'est assez habituel que les employés changent d'étage pour quelques dollars de plus. En 2003, le turnover pour des sociétés de sous-traitance à l'étranger est estimé à environ 15-20 %[34]. Pour l'anecdote, je l'ai vécu personnellement en 2013 lorsque je travaillais avec une société indienne. Sur une même année l'équipe a été changée quatre fois et, comme les préavis étaient inexistants, une perte de connaissance et surtout de temps s'est fait assez ressentir.

7.5.3. Différences culturelles

D'après Wikipédia[35], la majorité de la sous-traitance à l'étranger est réalisées dans des pays émergeants situés en Afrique, en Asie ou en Amérique du Sud. Or, en plus du problème de la langue, surviennent également des différences d'ordre religieuse, sociales. Toutes ces différences se reflètent évidemment dans la façon de collaborer et peuvent engendrer des risques non négligeables quant à la réussite d'un projet.

Il faut se rendre compte que la langue parlée qui a été apprise dans deux contextes différents, peut aussi créer un gap assez important car la façon dont les gens perçoivent le monde est différent.

[33] (Thiran, 2017)
[34] (Davinson, 2003)
[35] https://en.wikipedia.org/wiki/Offshore_outsourcing

7.6. Avantage de la sous-traitance à l'étranger et le développement Agile

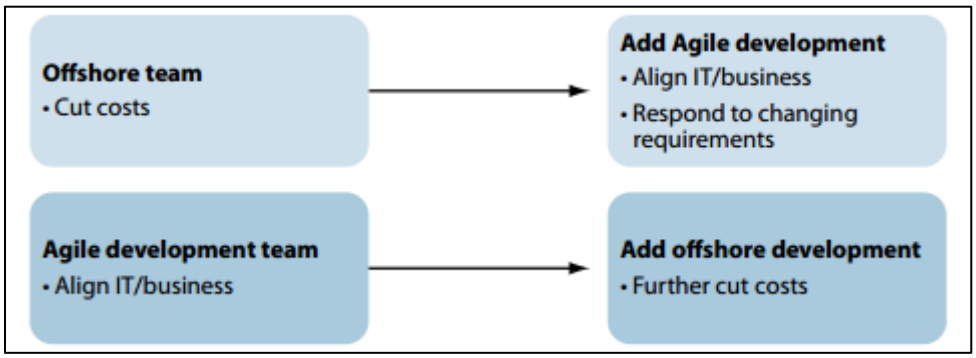


Figure 7 Gains lors de l'ajour de l'agilité et la sous-traitance à l'étranger[36]

Sous-traiter à l'étranger permet de réduire les coûts. Ajouter de l' « agilité » permet une meilleure flexibilité et ainsi répondre au changement plus rapidement et réduire les risques. En effet, comme on l'a vu précédemment, un des problèmes majeurs de la sous-traitance est le turnover important. Travailler par itération, plutôt qu'en waterfall, permet de réduire le risque de la perte d'une ressource au milieu du projet, et d'un rallongement du délai, étant donné qu'une partie de la solution aura déjà été déployée. De par la philosophie de collaboration des méthodes Agile, l'alignement entre le domaine IT et business est amélioré, ce qui permet également d'atténuer les risques spécifiques à la sous-traitance à l'étranger.

7.7. Problématique de l'agilité dans un environnement géographiquement distribué

L'approche Agile met en avant les valeurs comme les individus et leurs interactions, plus que les processus et les outils. D'ailleurs un de ces 12 principes issu des quatre valeurs vues précédemment est que les utilisateurs (ou leurs représentants) et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.[37]

Or, lorsque l'équipe de développement est géographiquement sous-traitée on rencontre les problèmes suivants :

- Problème de temps. Les intervenants ne sont pas tous situés sur un même fuseau horaire.
- Problème de langue. Les différents acteurs du projet ne parlent pas forcement la même langue et cela peut créer un gap assez important.
- Problème de culture. La façon de travailler et d'interagir avec les autres ne sont pas les mêmes suivant le continent, la religion.
- Problème de distance. Il n'est pas toujours possible de se voir physiquement.

[36] (Moore & Barnett, 2004)
[37] (Jeff Sutherland K. S., 2001)

Il est également important de souligner qu'il est parfois plus risqué de changer complètement l'organisation existante vers un environnement Agile plutôt que de sous-traiter à l'étranger via des méthodes de gestion de projet classique[38]

7.8. Solution proposée : Scrum

7.8.1. Justification du choix

Les méthodes Agiles promettent, contrairement aux méthodes traditionnelles, de garantir une meilleure satisfaction client, un taux inférieur de défauts, des livraisons rapides et une solution aux changements des spécifications client[39]. Cependant, ces méthodes ne sont appropriées qu'aux petits projets à faible complexité. Dans le cas de Beauteza vu la taille et la complexité du projet, cela s'applique parfaitement.

Il existe plusieurs méthodes Agile, mais celle qui est la plus connue pour gérer des projets Agiles dans le monde est la méthode SCRUM.

Sa principale caractéristique consiste à mettre en évidence les individus et la collaboration entre eux. Elle constitue une bonne solution aux changements, en tenant compte des processus, de la documentation complète, de la négociation des contrats et des plans préétablis. Soutenue par les piliers de la transparence, de la vérification et de l'adaptation, la méthode SCRUM vise à réduire les difficultés telles que le manque de planification, l'évolution constante des besoins, une portée mal définie, le manque d'investissement des clients et le manque de communication, communes à la plupart des projets. En outre, SCRUM est basé sur une livraison rapide, fréquente et continue de logiciels fonctionnels, une coopération continue entre le travail d'équipe et les affaires, l'excellence technique et la simplicité.

Toutes ces caractéristiques de SCRUM offrent des avantages considérables pour les chefs de projets ainsi que pour les clients finaux.

Qu'ils soient locaux ou distants, les membres doivent s'intégrer en tant que membre à part entière de l'équipe, car ils travaillent sur les mêmes engagements (exprimés lors de l'estimation) vis à vis du même SPRINT BACKLOG au sein de la même équipe SCRUM.

7.8.2. Scrum en bref

Scrum est une méthode Agile inventée pour la gestion de projet. Il s'agit d'un Framework qui a pour objectif d'améliorer la productivité d'une équipe :

[38] (Moore & Barnett, 2004)
[39] (Boehm & Turner, 2003)

7.8.2.1. Framework

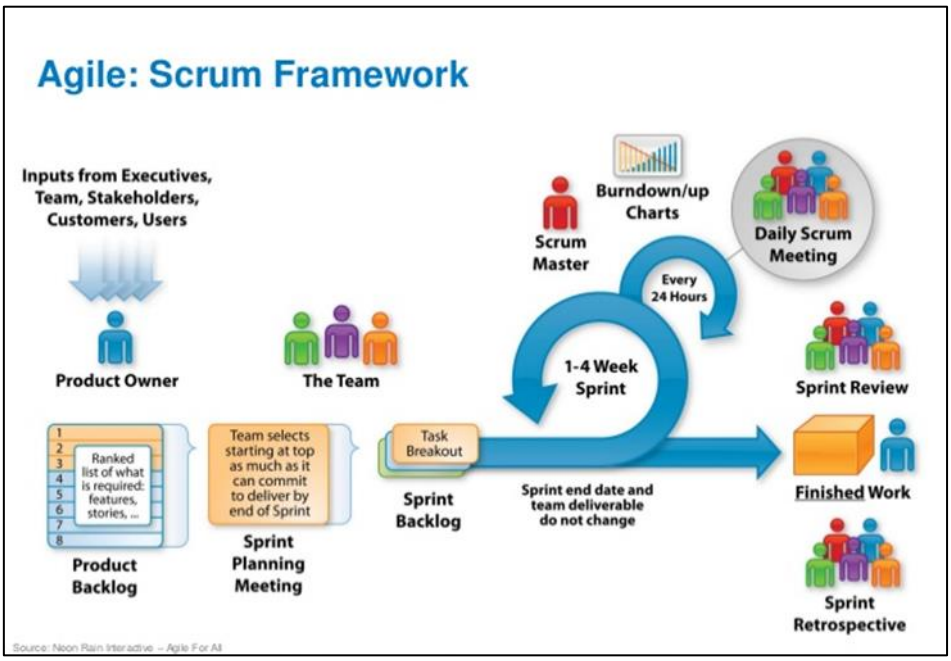


Figure 8 Scrum framework

Le framework Scrum se compose principalement des éléments suivants :

7.8.2.2. Rôles

Scrum définit seulement 3 rôles :

- Le Product Owner qui porte la vision du produit à réaliser et travaille en interaction avec l'équipe de développement. Il joue le rôle du client. Il a donc également un pouvoir de décision. C'est ce rôle-là que le Business Analyst doit tenir.
- L'Equipe de Développement qui est chargée de transformer les besoins exprimés par le Product Owner en fonctionnalités utilisables. Elle est pluridisciplinaire et peut donc encapsuler d'autres rôles tels que développeur, architecte logiciel, DBA, analyste fonctionnel, graphiste/ergonome, ingénieur système.
- Le Scrum Master qui doit maîtriser Scrum et s'assurer que ce dernier est correctement appliqué. Il a un rôle de coach à la fois auprès du Product Owner et auprès de l'équipe de développement. Il doit donc faire preuve de pédagogie. Il est également chargé de s'assurer que l'équipe de développement est pleinement productive. Généralement, le candidat tout trouvé au rôle de Scrum Master est le chef de projet. Celui-ci devra cependant renoncer au style de management « commander et contrôler » pour adopter un mode de management participatif.

7.8.2.3. Sprints

Le cycle de vie Scrum est rythmé par des itérations de quelques semaines cela s'appelle les sprints.

7.8.2.4. Artéfacts

7.8.2.4.1. Product backlog

Le product backlog contient les exigences initiales listées et priorisées avec le client. Il ne doit pas nécessairement contenir toutes les fonctionnalités attendues dès le début du projet étant donné qu'il va évoluer durant le projet en parallèle des besoins du client.

7.8.2.4.2. Sprint Backlog

Le sprint backlog contient les exigences issues du product backlog qui seront implémentée lors d'un sprint.

7.8.2.4.3. User stories

Les fonctionnalités décrites portent le nom d'user stories et sont décrites en employant la terminologie utilisée par le client.

Une User Story ou Story contient généralement les informations suivantes :

- ID – un identifiant unique
- Nom – un nom court (entre 2 et 10 mots), descriptif de la fonctionnalité attendue par le client (ex. Export / Import Standard Sales Item). Le nom doit être suffisamment clair pour que les membres de l'équipe et le Product Owner comprennent de quelle fonction il s'agit. Le nom ne doit pas introduire d'ambigüités.
- Importance – un entier qui fixe la priorité des Stories. La priorité d'une story peut être changée en cours de réalisation du projet.
- Estimation – La quantité de travail nécessaire pour développer, tester, et valider cette fonctionnalité. L'unité de mesure peut être un nombre de jours idéaux (jours à 100% dédiés à la fonctionnalité) ou un nombre de points. Les estimations se font en relatif en comparant les estimations des stories terminées avec la story à estimer.
- Demo – Un test relativement simple (ex : exporter un objet en XML puis l'effacer de la base, l'importer depuis le XML, à la fin l'objet doit être dans la base). Ce test constitue un test de validation.
- Notes – toute autre information : clarifications, références documentaires…

7.8.2.4.4. Burn up/down charts

Il s'agit d'un graphique qui donne une très bonne vision de ce qui a été fait et du rythme de travail de l'équipe. Il permet également d'anticiper si toutes les stories du Sprint Backlog seront terminées à la fin de l'itération ou non.

7.8.2.4.5. Le sprint planning meeting

On organise, avant chaque sprint, une réunion de planification : le sprint planning meeting. Ce planning sélectionne dans le product backlog les exigences les plus prioritaires pour le client. Elles seront développées, testées et livrées au client à la fin du sprint. Elles constituent le sprint backlog, un sous ensemble du product backlog.

7.8.2.4.6. Le daily scrum meeting

Au cours du sprint, il est organisé, chaque jour, une réunion d'avancement (environ 15 min) avec tous les membres de l'équipe, afin de s'assurer que les objectifs du sprint seront tenus : c'est le Scrum ou mêlée en français. Chaque jour, après la réunion Scrum, le Scrum Master maintient le burndown chart.

Cette réunion n'a pas seulement un but purement informatif, mais a aussi pour but de stimuler l'esprit travail en équipe et le niveau d'engagement de chaque membre de l'équipe dans le projet. Durant la réunion, chaque membre de l'équipe doit prendre la parole et présenter principalement les choses suivantes :

- Ce que j'ai fait hier et les éventuels problèmes rencontrés
- Ce que je vais faire aujourd'hui
- Est-ce que j'ai des difficultés pour continuer mon travail
- En faisant cet exercice quotidiennement chaque membre de l'équipe est au courant de ce que font ses collègues et il peut coordonner son travail et aider ou se faire aider en cas de difficultés

Le Scrum Meeting n'est pas une réunion pendant laquelle on cherche à résoudre les problèmes, mais uniquement à les identifier et les exprimer. Le Scrum Master a pour rôle d'apporter des solutions ou de déléguer à un autre membre de l'équipe la résolution des problèmes soulevés durant le Scrum Meeting. A la suite de cette réunion le Scrum Master met à jour le burndown chart.

7.8.2.4.7. Sprint review meeting

A la fin d'un sprint, on fait une démonstration au client des derniers développements, c'est le Sprint Review Meeting. C'est aussi l'occasion de faire un bilan, sur le fonctionnement de l'équipe et de trouver des points d'amélioration.

7.8.2.4.8. Spring retrospective meeting

C'est un meeting d'environ trois heures avec l'équipe de développement. Il s'agit de répondre aux trois questions suivantes :

- Qu'est ce qui a bien été dans ce sprint
- Qu'est ce qui peut être amélioré
- Qu'est-ce qu'on va améliorer pour le sprint suivant

C'est un meeting très important car cela permet à l'équipe de mieux s'organiser pour le sprint suivant et donc augmenter sa vélocité.

7.9. Difficulté et blocage de la méthodologie SCRUM avec une équipe offshore et solutions envisagées

Comme nous l'avons vu précédemment, utiliser une méthode Agile dans des projets distribués amène des reconsidérations. Pour le framework Scrum tel qu'expliqué au chapitre précédent cela ne fonctionnera également pas étant donné que les équipes sont réparties à différents endroits géographiques et ne sont donc pas physiquement en contact. Il faut des adaptations pour utiliser ce framework et réduire le risque lié à la distance. Les points suivants doivent donc être absolument étudié avec attention avant de commencer.

Tout d'abord il existe trois types de configuration possible [40]:

- **Equipes Scrum isolées** : Les équipes Scrum sont isolées les unes des autres géographiquement.
- **Les Scrum de Scrum distribuées** : Les équipes Scrum sont isolées les unes des autres et intégrées par un Scrum de Scrum qui se réunit régulièrement.
- **Equipes Scrum distribuées** : Les équipes Scrum sont cross fonctionnelles et distribuées au travers des continents, où chaque équipe Scrum a des membres à plusieurs endroits.

Malgré que la dernière configuration soit la plus efficace [41] dans le cadre de ce mémoire je m'attarderai cependant uniquement que sur le premier cas, car c'est dans cette configuration que l'on va travailler pour le projet Beauteza.

7.9.1. Choix d'une équipe

Une équipe mature ayant fait ses preuves doit être choisie afin de lier un partenariat de confiance. Il est exclu de sélectionner une équipe ayant peu d'expérience en SCRUM vu les contraintes engendrées par l'outsourcing.

L'idéal pour commencer serait une équipe hybride avec seulement 50 % des effectifs à l'étranger. Il est également utile, dans la mesure du possible, d'organiser des rencontres physiques.

[40] (Jeff Sutherland G. S., 2008)
[41] (Jeff Sutherland G. S., 2008)

7.9.2. Choix d'un leader technique – point de contact et proxy product owner

Il est nécessaire de nommer au sein de l'équipe distante une personne assez expérimentée qui sera le point de contact avec l'équipe locale et le product owner pour toutes questions techniques[42]. Cependant, cela fonctionne si le product owner est disponible 100 % du temps et que les équipes travaillent sur le même fuseau horaire. Dans de nombreux projets, et notamment dans le cas de Beauteza, le product owner n'est souvent pas complétement 100% disponible, et surtout, pas au même moment que l'équipe de développement. Il faut donc nommer une personne qui joue le rôle de proxy product owner qui comble le manque de disponibilité du product owner. Ils travaillent tous les deux en « binôme », avec les mêmes responsabilités, à l'exception que le proxy owner n'a pas de pouvoir de décision.

7.9.3. Expérience et autorité du scrum master

Le scrum master doit avoir un haut niveau de compétence, de qualifications et d'expérience pour ce rôle. De plus idéalement il doit pouvoir parler les langues des deux équipes[43].

Son rôle est donc essentiel pour la réussite du projet. Ses qualités doivent donc être :

- Connaissance approfondie de la méthodologie Scrum,
- Capacités à savoir résoudre les conflits et les problèmes,
- Avoir des facilités de présentation, de communication et de négociation,
- Savoir guider sans imposer, être un coach plutôt qu'un chef de projet
- Savoir communiquer de façon précise sur le degré d'avancement,
- Garder le respect de l'objectif essentiel, qui est de livrer un produit qui apporte de la valeur.

7.9.4. Problèmes de la culture

Il est important, en début de projet, de prendre en compte l'aspect culturel de l'équipe offshore avec laquelle on va travailler. Beaucoup de variables ne sont jamais prise en compte car les personnes pensent, à tort, qu'au niveau du travail et des priorités, les notions sont les mêmes. Or, par exemple, de nombreux développement sont fait en Inde sans tenir compte de la notion de caste. Et cet aspect culturel est primordial. Dans le cas de Beauteza, la notion du temps et de la ponctualité n'était pas perçue de la même manière entre les deux équipes.

Il est donc important de mener une analyse sur cet aspect-là, avant de même commencer le projet, et de l'inclure dans celui-ci.

Si c'est possible il est vivement conseillé de rencontrer l'équipe et de travailler sur place quelques semaines, ou inversément, pour s'imprégner de la culture[44].

[42] (DeYoe, 2009)
[43] (DeYoe, 2009)
[44] (Jeff Sutherland G. S., 2008)

Il existe un modèle d'analyse de la culture nationale développé par Geert Hofstede qui permet de comparer les cultures de différents pays sur 6 dimensions[45]. Il s'agit du pouvoir (égalité contre inégalité), du collectivisme (par opposition à l'individualisme), de l'évitement de l'incertitude (par opposition à l'acceptation de l'incertitude), de la masculinité (par opposition à la féminité), de l'orientation temporelle et du plaisir (par opposition à la modération). Hofstede a réuni la plupart de ses données sur les valeurs culturelles mondiales par le biais d'enquêtes menées par IBM dans plus de 50 pays. Il proposa ensuite un barème utilisant une échelle de 1 à 120 suivant les dimensions[46]. Examinons en détails ces dimensions comme le décrit Hofstede dans son ouvrage :

7.9.4.1. L'index de distance par rapport au pouvoir

« La distance par rapport au pouvoir consiste en l'acceptation et l'attente, par les membres des organisations et des institutions ayant le moins de pouvoir, de ce que le pouvoir soit distribué de manière inégale » selon Hofstede. Cette première dimension ne mesure pas le niveau de distribution du pouvoir dans une culture donnée, mais ce que la perception de la population. Un score faible de distance par rapport au pouvoir indique qu'une culture attend et accepte que les relations de pouvoir soient démocratiques et que ses membres soient perçus comme égaux. A l'opposé un score élevé signifie que les membres de la société qui dispose de moins de pouvoir sont conscient de l'existence d'une hiérarchie.

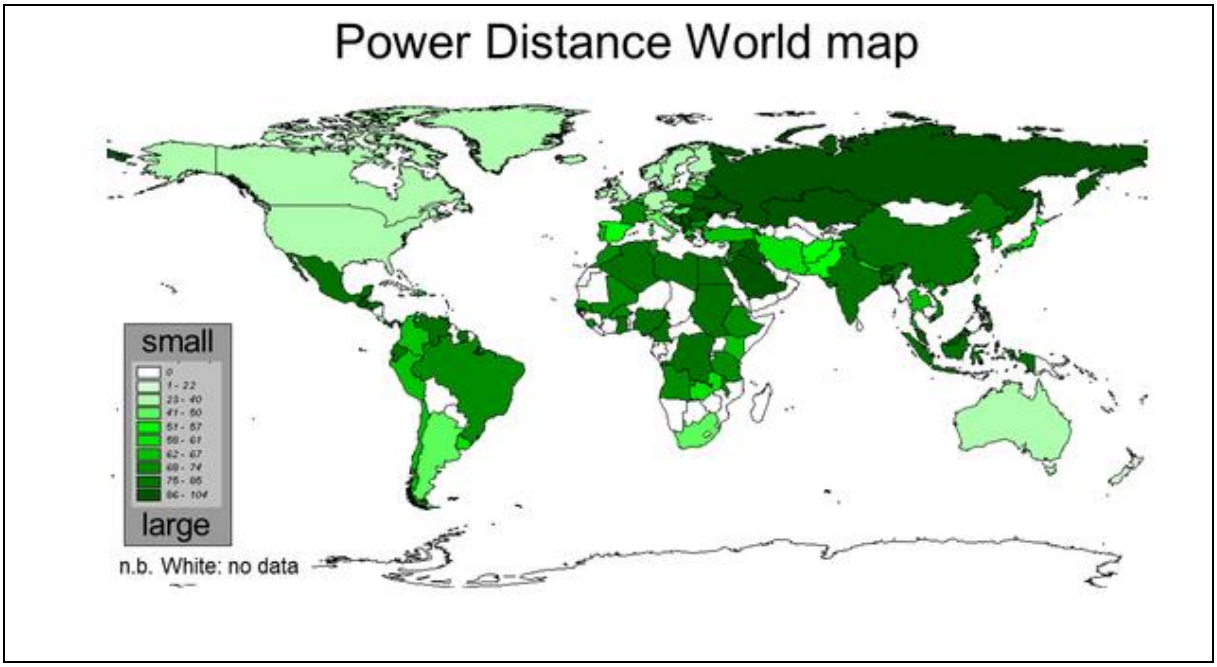


Figure 9- distance – pouvoir

[45] (Hofstede, s.d.)
[46] (Hofstede, Hofstede, & Minkov, Cultures and Organizations, Software of the Mind. Third Edition, 2010)

7.9.4.2. Individualisme

Cette dimension mesure « Le degré auquel les individus sont intégrés aux groupes. ». Les cultures individualistes donnent de l'importance à la réalisation des objectifs personnels, tandis que dans les sociétés collectivistes, les objectifs du groupe et son bien-être ont plus de valeur que ceux de l'individu lui-même.

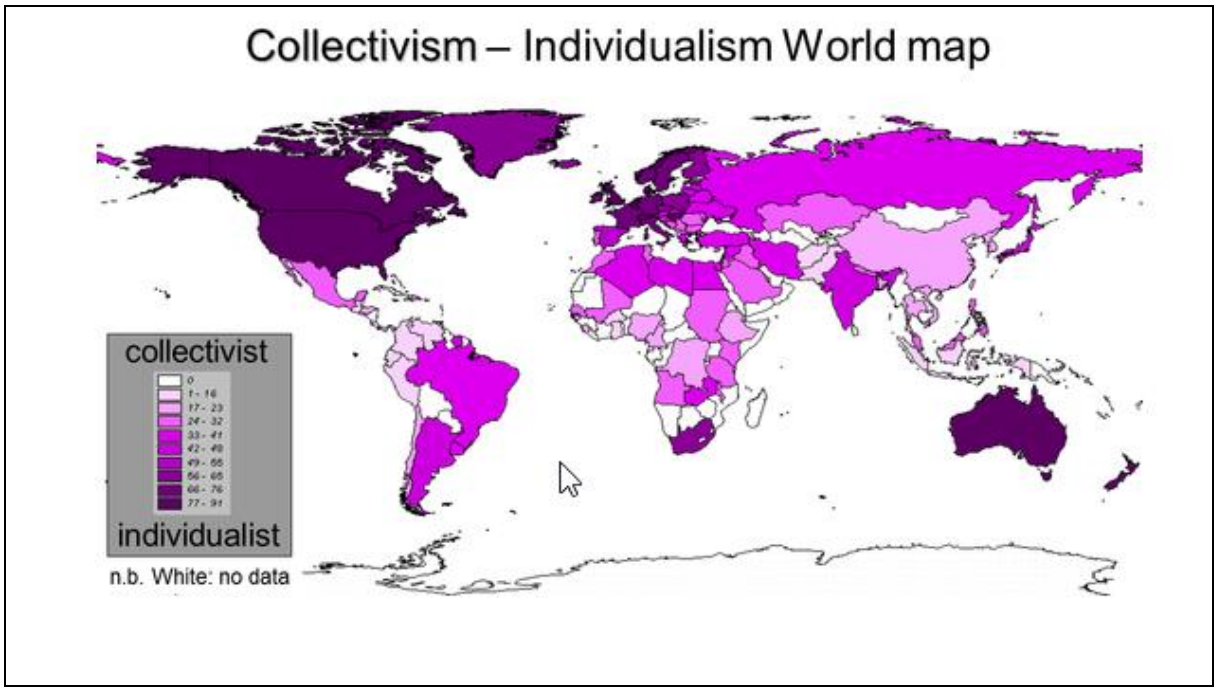


Figure 10 – Individualisme

7.9.4.3. Indice évitement-incertitude :

« La tolérance d'une société pour l'incertitude et l'ambiguïté. ». Cette dimension mesure la façon dont une société gère les nouvelles situations, les évènements inattendus et le ressenti face à un changement. Les pays qui ont un score élevé sont plus résistants au changement et ont tendance à imposer des règles rigides, des règlements et/ou des lois pour minimiser le risque tandis que les pays dont l'indice est faible s'adaptent mieux au changement et disposent de moins de règles pour formaliser celui-ci.

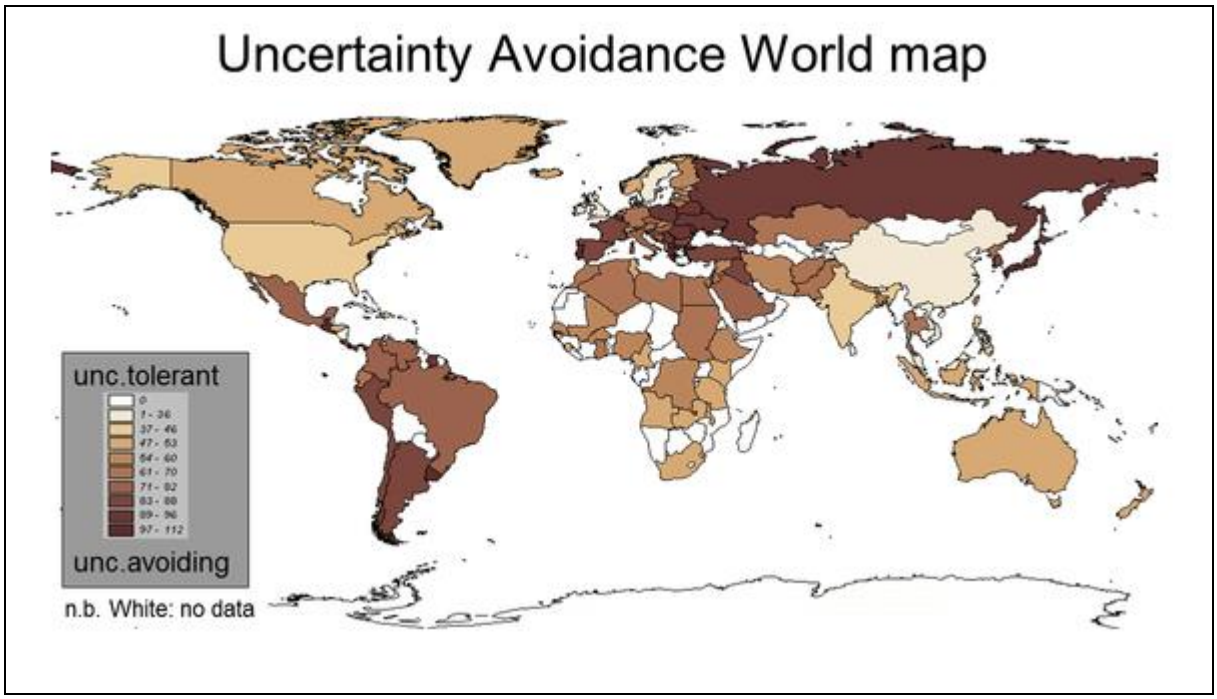


Figure 11 – Incertitude

7.9.4.4. Masculinité

« La distribution des rôles émotionnels entre les genres. » Cette dimension mesure le niveau d'importance qu'une culture accorde aux valeurs « masculines » telles que l'assurance, l'ambition, le pouvoir et le matérialisme, ainsi qu'aux valeurs « féminines » telles que les relations humaines. Les pays avec un score élevé présentent généralement des différences plus évidentes entre les genres et ont tendance à être plus compétitifs et ambitieux tandis que ceux dont le score est bas, présentent moins de différences entre les genres et accordent plus de valeur aux relations.

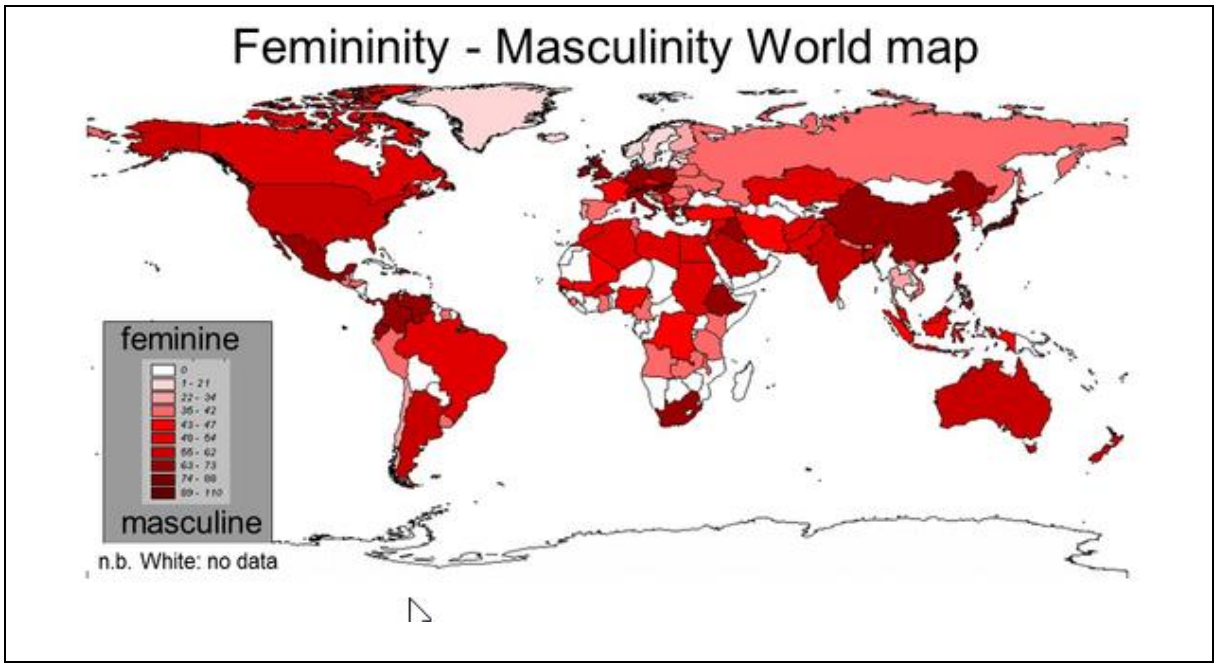


Figure 12 - Masculinité

7.9.4.5. Orientation à long terme

Cette dimension décrit si un pays a plutôt une vision long terme ou court terme par rapport à ses besoins. Les pays avec une vue court terme donnent de la valeur aux méthodes traditionnelles, prendre un temps considérable pour créer des relations. Pour les pays avec une orientation à long terme regarde le futur plutôt que le présent ou le passé. Une telle société vise plutôt des objectifs et donne de la valeur aux récompenses.

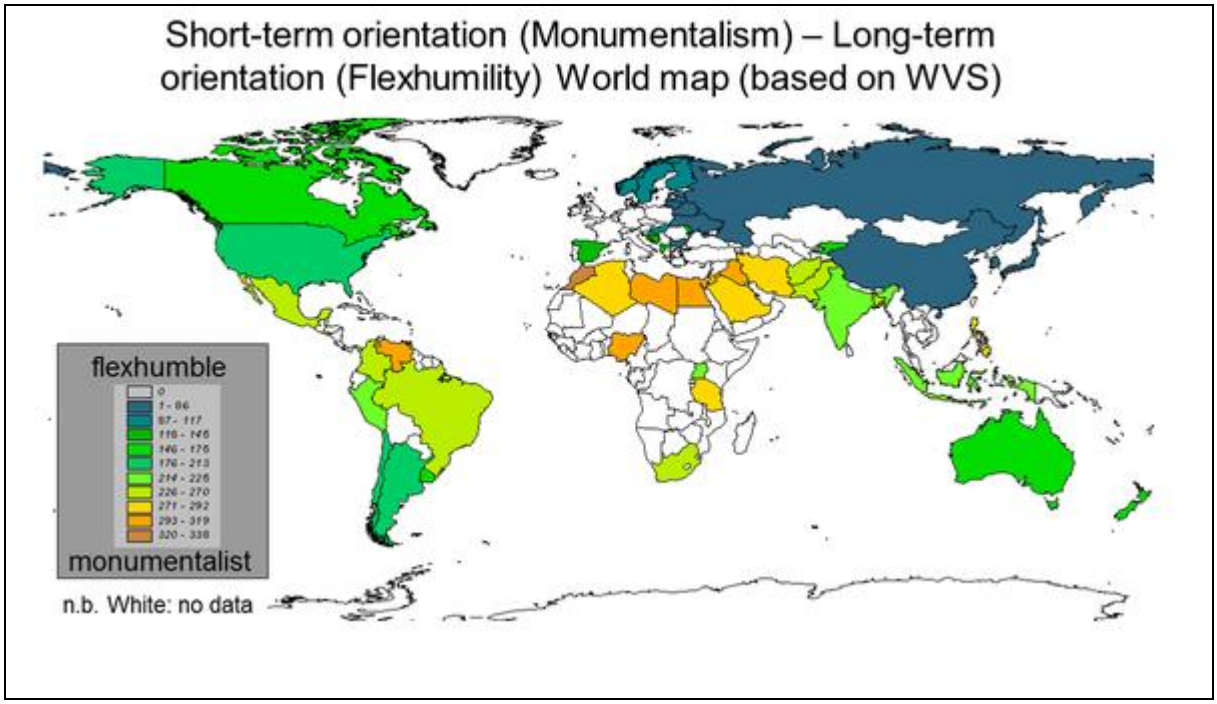


Figure 13 - Orientation long-court terme

7.9.4.6. Indulgence :

Cette dimension mesure la capacité d'une culture à satisfaire les besoins immédiats et les désirs personnels de ses membres. Les cultures donnant de la valeur à la modération disposent de règles sociales strictes et de normes en dessous desquelles la satisfaction des pulsions est régulée et découragée.

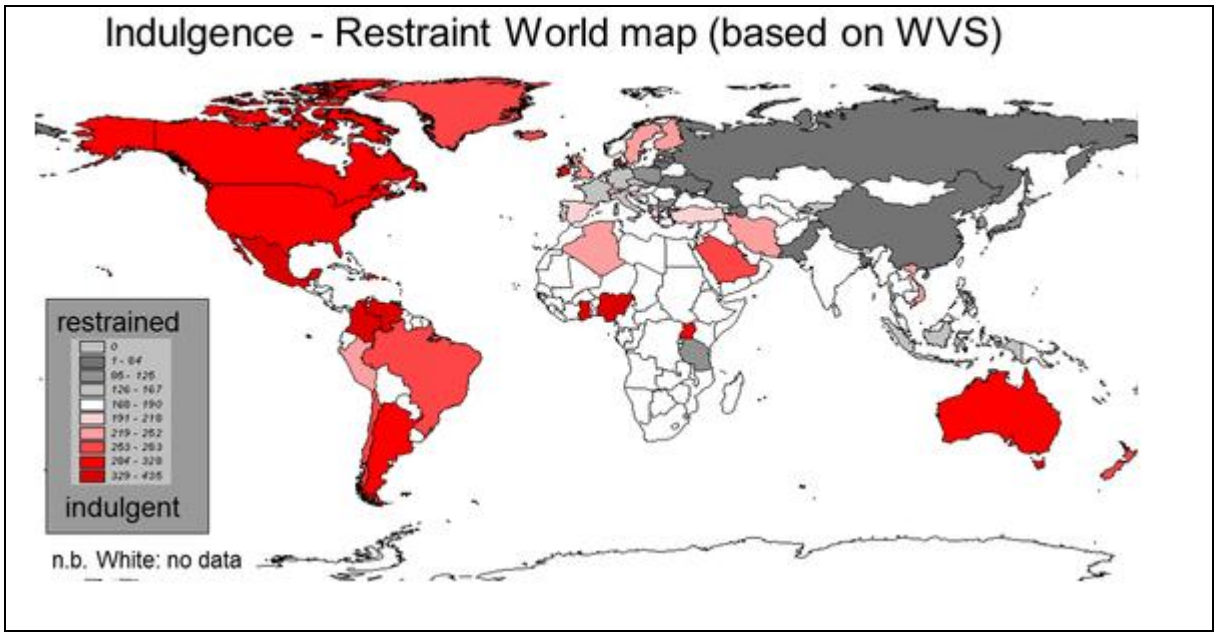


Figure 14 - Indulgence

Si nous comparons en détails maintenant le Chili et la Belgique grâce à l'application disponible sur https://geert-hofstede.com nous pouvons y voir des différences importantes au niveau de l'individualisme et la vision long terme entre ces deux pays.

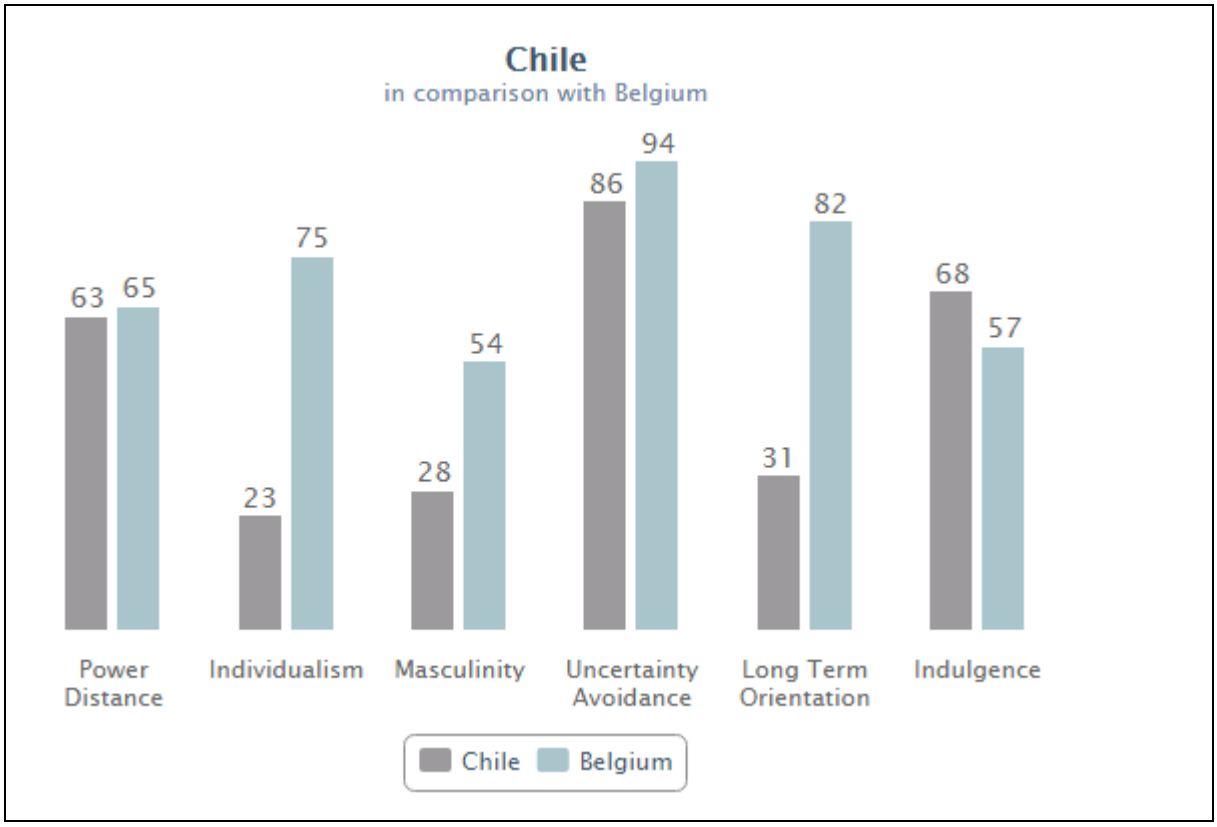


Figure 15 - Comparaison Chili - Belgique

Sur base de ces résultats, il s'avère effectivement plus difficile de mettre place une gestion de projet waterfall avec des objectifs à long termes et des milestones. Des sprint Scrum, par définition très courts, semblent plus appropriés. De plus, Agile prônant les relations avec les individus plutôt que les documentations, et les chiliens n'étant pas très individualistes et masculins, le choix de cette méthodologie s'avère plus judicieux. Cependant on observe également une distance hiérarchique assez forte et une volonté de tout contrôler avec des procédures assez lourdes comme en Belgique, ce qui n'est parfois pas toujours facile à concilier avec des méthodes Agile.

7.9.5. Problèmes de communication

La communication, spécialement avec les développeurs, est un challenge quand l'équipe de développement et le business ne sont pas localisés au même endroit. L'interaction entre les individus tel que prônée par Scrum est beaucoup plus compliquée à mettre en place.

Dans un tel projet, il paraît également clair que l'email seul ne suffit pas comme moyen de communication.

De plus, un autre problème est que le langage écrit ne reflète pas les émotions et un même message peut être interpréter de façon différente suivant les personnes. Il faut donc multiplier les canaux de communication afin de briser cette barrière. Des solutions de vidéo conférence comme Skype[47] ou Zoom[48], par exemple, qui permettent en plus l'enregistrement de ces conférences, permettent de répondre à ce besoin.

Il faut également mettre en place des méthodes et des standards pour le partage des différents documents. L'implémentation d'un site de collaboration commun (par exemple Slack[49]) pour l'historique des échanges, et d'un repository (sharepoint par exemple) dans lequel se trouve tous les artéfacts critiques du projet, est nécessaire. Cela permet à chacun de travailler sur la même base Il faut également un outil de management Scrum (Scrumwise par exemple) dans lequel tous les utilisateurs enregistrent leurs progressions. Cela permet d'alimenter de manière précise le product backlog, sprint backlogs et Burn up/down chart sur une base quotidienne[50].

Il est également important d'avoir une vue claire sur l'état d'avancement continu du projet. C'est pour cela que les daily scrum meeting sont primordiaux. Il faut absolument un contact visuel : l'équipe locale face à l'équipe distante. La vidéo doit montrer les équipes dans leurs ensembles, le déroulement est le même que pour un daily sprint meeting avec une équipe locale, chacun parle à son tour.

Il faut également adopter une langue et une terminologie commune. Un mot n'a pas la même signification pour tous. Il faut bien s'assurer que chaque mot a la même signification et est compris de la même façon par chacun. Pour ce faire il peut être utile d'élaborer un dictionnaire/glossaire[51].

Il faut toujours s'assurer que tous les partis ont bien compris le même message. Il ne faut pas hésiter à reformuler si nécessaire[52].

[47] https://www.skype.com
[48] https://zoom.us/
[49] https://slack.com/
[50] (DeYoe, 2009)
[51] (Englebert, 2017)
[52] (Maquet, 2015)

Une démarche d'amélioration continue est également nécessaire. Les rétrospectives à chaque fin de « sprint » permettent de remonter les suggestions d'amélioration.

Un « leader » doit également être nommé de façon à avoir un seul point de contact en cas de divergence. L'idéal si le budget le permet, est d'avoir un représentant de l'équipe off-shore dans l'équipe on-shore. Cela a pour but de faciliter la communication.

Bien qu'un des principe Agile soit de privilégier la communication plutôt qu'une documentation exhaustive, en cas de sous-traitance il est malheureusement nécessaire d'avoir plus de documentation que pour un projet local. Etant donné que les équipe travaillent en « remote », il est conseillé d'utiliser un outil de versionning efficace de ces documents afin de comparer les différentes versions et d'avoir la dernière version mise à jour[53].

7.9.5.1. Problématique des users stories

Comme on l'a vu précédemment une user Story est courte, discutée et confirmée par des tests d'acceptation rédigés au même moment que celle-ci. Elles favorisent donc le mode oral et collaboratif.

Cependant le langage courant utilisé dans ces users stories peut amener à des interprétations ambiguës, et la distance et les langues différentes parlées par les interlocuteurs compliquent encore plus la situation. Comme on l'a vu précédemment la langue parlée qui a été apprise dans deux contextes différents, peut créer un gap assez important car la façon dont les membres de l'équipe perçoivent le monde est différent. En effet, dans le cas de Beauteza malgré que l'espagnol soit la langue commune, l'espagnol du Chili, celui de Colombie et celui d'Espagne que j'ai appris comportent des différences assez importantes au niveau de la prononciation, de la grammaire et du vocabulaire.

Il faut donc compléter ces users stories avec un langage de modélisation adéquat. UML[54] répond à ce critère.

7.9.5.2. UML

Trop souvent les spécifications sont rédigées dans de simple documents de type Microsoft office et celles-ci peuvent être interprétées de façon différente selon les individus. En effet les langues parlées sont par définition des langages déterministes, il se peut que la perception d'une même phrase puisse être comprise différemment par un interlocuteur. C'est pourquoi il est nécessaire d'utiliser un langage formel universel qui ne soit pas ambigu dans le cadre de la rédaction de ces spécifications et particulièrement pour compléter les user stories. Cependant un seul modèle n'est pas suffisant pour couvrir toutes les vues à travers une spécification complète[55] :

[53] (Fowler, 2006)
[54] http://www.uml.org/
[55] (Kolp, 2016)

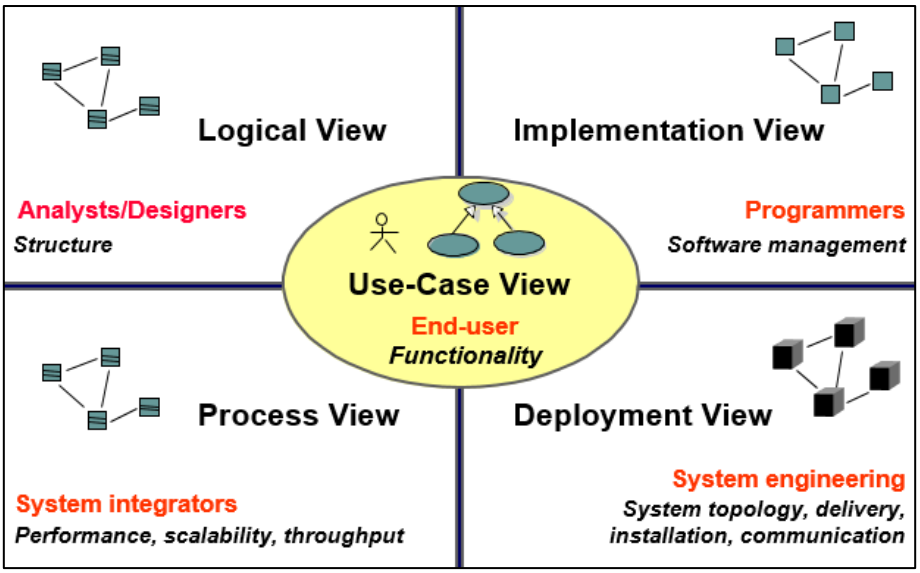


Figure 16 Les quatre vues d'une spécification

Le langage UML répond à cette problématique. *UML* est le langage standard de facto pour la conception de logiciels. C'est une norme qui est enseignée dans de nombreuses écoles et universités à travers le monde et de plus c'est un standard adopté par l'OMG[56] et donc validé par de nombreuses entreprises.

UML permet entre autres de[57] :

- Obtenir une modélisation de très haut niveau indépendante des langages et des environnements
- Faire collaborer des participants de tout horizon autour d'un même document de synthèse
- Exprimer dans un seul modèle tous les aspects statiques, dynamiques, juridiques, spécifications, etc…
- Documenter un projet
- Générer automatiquement la partie logiciel d'un système

UML dispose de 14 types de diagrammes différents ce qui en fait un langage complexe à utiliser. Cependant, dans la pratique, tous ces diagrammes ne sont pas utilisés pour chaque projet. Les diagrammes les plus utilisés dans la pratique sont les use case, les diagrammes de séquences et les class diagram. Pour être opérationnel et efficace avec UML il n'est donc pas nécessaire de maîtriser l'ensemble des diagrammes que le langage propose. Bien qu'UML soit plus approprié dans des projets de type waterfall, et particulièrement utilisé avec la méthodologie RUP[58], il peut être également utilisé dans le cas de Scrum. Normalement Scrum ne recommande aucunes méthodologies particulières car il favorise l'interaction entre les individus. Cependant dans le cas bien précis de la sous-traitance à l'étranger, UML s'avère très pratique et évite les malentendus.

Dans le cas de la rédaction des spécifications au niveau de Scrum, les use cases peuvent remplacer les users stories, il faudra cependant veiller à ce qu'ils soient réalisables en une itération, quitte à le compléter par la suite pour le sprint suivant.

[56] http://www.omg.org/
[57] (Roques, 2009)
[58] (Kruchten, 2000)

Dans le cas d'un use case complexe qui possède un cas nominal et plusieurs cas alternatifs il peut également être divisé en plusieurs scénarios, chaque scénario correspondant à un cas. A ce niveau on se rapproche de la définition d'une user story.

Les tests d'acceptation devront également vérifier ces uses cases.

L'activity et le sequence diagram peuvent également compléter chaque spécification pour chaque sprint, de façon à vérifier l'enchainement.

Cependant il faut bien tenir compte du fait que dans le cas d'un projet Scrum les diagrammes peuvent évoluer régulièrement, et les maintenir peut prendre du temps.

7.9.5.3. Partage du contexte et des priorités

Un dernier point dans la communication est le Partage du contexte et des priorités. Dans un contexte de développement distribué, il est parfois difficile de communiquer à l'équipe Offshore les nuances du contexte et des priorités du client. Afin de distribuer activement une telle connaissance du contexte client, il est important de prévoir des voyages réguliers dans les deux sens si le budget le permet, des connections Skype permanentes, la mise en place d'une « gazette projet » envoyée à la fin de chaque itération et des contacts informels avec le Product Owner[59].

7.9.6. Notion de « terminé »

Un des problèmes majeurs lors du développement initial est que la notion de terminé n'était pas comprise par l'équipe de développement et le Business Analyst. Il est nécessaire de passer du temps à échanger sur la définition de « Terminé » pour les « stories » et « sprints ».

Il est important d'avoir un document qui indique les points nécessaires à vérifier avant la livraison. Il est primordial que ce document soit bien compris par les 2 parties. Comme vu précédemment, la barrière de la langue et de la distance étant là, il est nécessaire que chaque partie puisse expliquer en reformulant avec ses mots si nécessaires lors du meeting pour la planification du sprint malgré que cela puisse prendre du temps.

Les critères d'acceptation sont également décrits dans les user stories.

7.9.7. Gestion, planification du temps et organisation

Un autre gros problème lors d'un projet avec une équipe offshore est l'organisation des journées. En effet, ne travaillant pas forcément sur le même fuseau horaire, et donc au même moment, il est primordial de bien se synchroniser.

[59] (Jeff Sutherland G. S., 2008)

Une solution envisagée est de décaler son horaire journalier de quelques heures pour chacune des parties afin d'être alignés le plus d'heure possible sur une journée et ne pas attendre le lendemain pour obtenir une réponse[60].

De plus, il est important, comme nous l'avons vu précédemment de tenir compte et d'intégrer la culture des personnes. La notion de temps et d'heure précise n'est pas perçue de la même façon suivant celle-ci. Pour certaines cultures une réunion doit commencer à une heure précise tandis que pour d'autre il s'agit plutôt d'une tranche horaire de la journée par exemple.

Il est nécessaire également de définir dès le début un cadre et les priorités de chacun, avec des responsabilités clairement définies. Plusieurs exemples[61] montrent que cela a bien fonctionné.

7.9.8. Définition des sprints

Un exercice délicat est la priorisation des requirements pour les sprints. Pour ce faire la matrice MoSCoW définie précédemment s'applique parfaitement à ce cas de figure. Il devient alors aisé de choisir les requirements les plus importants en premier lieu dans le product backlog afin de constituer un sprint.

7.9.8.1. Durée d'un sprint

Habituellement, pour des projets locaux, la durée est de 4 semaines. Cependant, pour des projets à l'étranger, réduire le temps de 1 à 2 semaines, du moins au début, s'avère plus efficace pour les corrections, pour le respect du planning et pour le feedback des utilisateurs[62].

Il est important également de faire travailler toute l'équipe sur une story à la fois et s'assurer que le cycle de livraison est le plus court possible afin qu'il soit possible de recevoir un feedback rapide des questions du client[63].

Une fois que l'équipe est rodée et que la confiance est établie on peut éventuellement rallonger la durée du sprint jusqu'à 4 semaines comme le recommande Scrum.org.

7.9.9. Vérification continue de la qualité

L'objectif de cette étape est de détecter les bugs le plus vite possible et le plus tôt possible.

Un des problèmes majeurs de la sous-traitance à l'étranger est que le travail est en « fixed price ». L'équipe de développement est facturée pour une release ou une fonctionnalité. Dès lors la qualité passe en second lieu car le plus important est de livrer quitte à laisser des bugs ou des spécificités incomplètes qui seront corrigées lors de la maintenance ou via un change request.

[60] (DeYoe, 2009)
[61] Pawel Mysliwiec, Scrum.org
[62] (DeYoe, 2009)
[63] (Moore & Barnett, 2004)

Il est donc important à chaque itération, d'établir des jeux de tests fiables, complets et validés afin de vérifier tout au long du projet que la qualité est présente. Les tests doivent être vérifiés à chaque sprint, car ces problèmes sont en effet beaucoup plus coûteux à trouver et à réparer après le déploiement.

La qualité est l'affaire de toute l'équipe, et une stratégie de tests efficaces doit être également définie en début de projet.

L'objectif est évidemment de tester le produit tout le long du sprint et pas uniquement à la fin.

Il faut donc établir au début de chaque Sprint :

- Les tests de non régression qu'il faut exécuter à chaque fin de sprint, ainsi que pendant le sprint, par l'équipe de développement. Ces tests peuvent être éventuellement automatisés.
- Les tests unitaires qui seront réalisés. Pour ce faire les critères d'acceptation sont également décrits dans les user stories. Pour ce faire chaque use case UML devra être décrit de manière explicite de la façon suivante[64] :

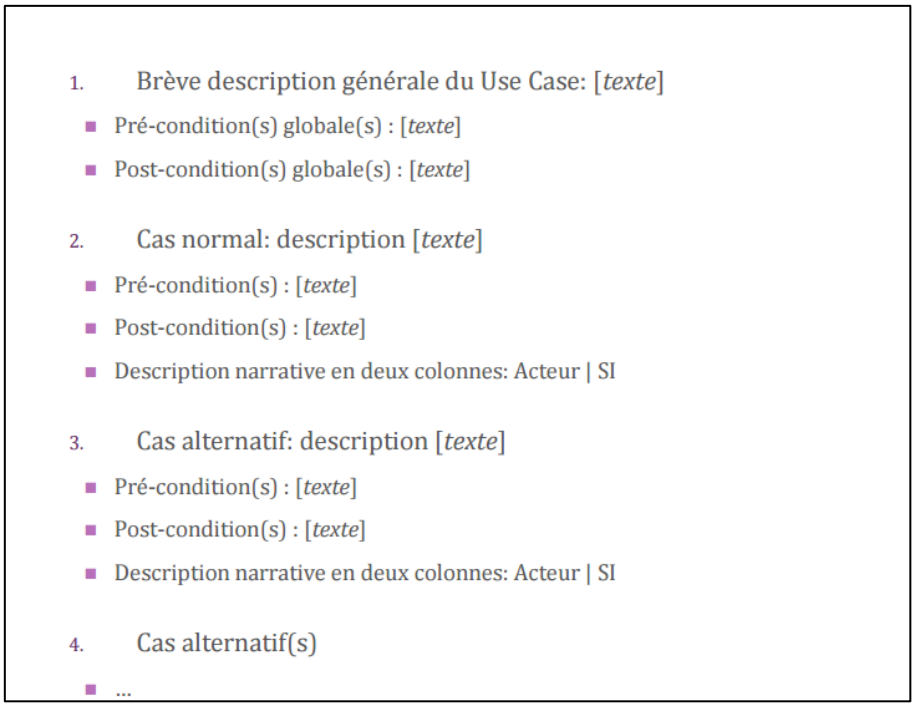

1. Brève description générale du Use Case: [*texte*]
   - Pré-condition(s) globale(s) : [*texte*]
   - Post-condition(s) globale(s) : [*texte*]
2. Cas normal: description [*texte*]
   - Pré-condition(s) : [*texte*]
   - Post-condition(s) : [*texte*]
   - Description narrative en deux colonnes: Acteur | SI
3. Cas alternatif: description [*texte*]
   - Pré-condition(s) : [*texte*]
   - Post-condition(s) : [*texte*]
   - Description narrative en deux colonnes: Acteur | SI
4. Cas alternatif(s)
   - ...

Figure 17 Description d'un use case

L'équipe de développement devra donc vérifier chaque use case développé avant la livraison.

7.9.10. Métriques pour mesurer l'avancement

En plus du burndown chart, afin de garder une vue optimiste sur l'avancement des sprints et la qualité de développement de l'équipe, il est conseillé de définir également des métriques. Elles peuvent être par exemple :

- Mesure du nombre de bugs trouvés par release
- Mesure du nombre de bugs résolus par release

[64] (Burnay, 2016)

- Mesure du nombre de bugs par priorité, par fonction
- Résultats des tests
- Complexité du code
- Couverture du code par les tests
- Pourcentage des « stories » terminées par sprint

7.9.11. Une partie du travail doit se faire localement

Même si l'ensemble des développements peuvent être distribués, il s'avère souvent que certaines tâches du projet peuvent l'être difficilement et doivent donc être traitées localement. Des exemples de tâches locales sont : écrire de la documentation dans la langue de l'utilisateur, se coordonner avec les responsables de l'Architecture du client, discuter des spécifications techniques et ce genre de choses

7.9.12. Mise en production

Une fois un sprint terminé, on peut envisager sa mise en production ou non. Si on choisit de le faire il faut organiser un déploiement progressif plutôt qu'un total sur le marché. Il est nécessaire d'avoir un panel d'utilisateur pilote et de superviser ceux-ci. L'intérêt est de créer du feedback très tôt, de réaliser des tests de volumétrie, ainsi que d'avoir un panel plus hétérogène d'utilisateurs qui génèrent des cas réels.

Dans un premier temps, un déploiement peut être exécuté vers des utilisateurs de l'entourage proche de la société. Cette première étape servira à valider l'application dans un environnement réel et éventuellement revoir les besoins pour les futurs sprints.

Ensuite une fois que celle-ci est validée, un second déploiement vers des utilisateurs pilotes sélectionnés au préalable est exécuté. La durée de ces tests doit être définie en début de projet et un accompagnement à ces utilisateurs doit être réalisé.

Pour Beauteza une chaine de salon de beauté à Santiago a été choisie comme pilote pour l'application.

Une fois ces deux étapes validées le roll out peut être organisé. Concernant cette étape en fonction de la complexité et de l'impact d'un nouveau logiciel, une conduite au changement devra peut-être être réalisée afin de garantir la réussite du projet[65]. Dans le cas de Beauteza il s'agit de mettre en place des formations pour les gérants des salons de beauté afin qu'ils utilisent l'application de manière efficace.

[65] (Dejean, 2016)

7.9.13. Maintenance

Après une mise en production réussie il est également important de se préoccuper de la phase de maintenance avec le partenaire de confiance.

La maintenance est la phase qui intervient après la période de garantie. Le fonctionnement de celle-ci diffère totalement de la phase projet. Une fois que le projet est en production une résolution rapide des erreurs est primordiale. Il est donc important de définir des procédures efficaces.

L'intérêt de ces procédures est de simplifier les échanges et d'éviter les allers-retours permanents en cas d'anomalie, ou de change request, entre les demandes client, l'équipe commerciale de la société de sous-traitance et l'équipe de développement.

De plus, dans le cas d'une urgence, ce processus serait trop long et pourrait poser des problèmes et des pertes de temps importantes.

Pour cela, chaque demande de correction d'anomalie doit être définie dans une procédure précise : on la fait passer dans une évaluation permettant de définir son degré d'urgence et ainsi les délais de correction que le prestataire s'engage à tenir (SLA). Les processus d'« escalation » doivent être également clairement définis.

Pour ce se faire on peut s'inspirer des principes d'ITIL – DSS02 : Manage Service Requests and Incidents[66].

Concernant les change request ceux-ci font l'objet d'une estimation du coût et d'une planification.

Dans la pratique il faut réaliser plusieurs étapes. Tout d'abord, il faut définir une équipe de support avec un responsable attitré. Ensuite, il faut s'accorder sur la façon dont les utilisateurs peuvent remonter des anomalies. Cela peut être par le biais d'une ligne téléphonique, d'une adresse email ou via un site internet. Enfin, un système de gestion de tickets doit-être mis en place afin d'assurer le suivi. Pour chaque anomalie reportée, un ticket est créé et une priorité est définie. Il est important également de définir des procédures pour assurer une communication efficace avec le client qui a contacté le support technique et lui donner un suivi approprié.

[66] (Wautelet, 2016)

8. Conclusion

Le rôle d'un Business Analyst dans un projet informatique est important. De nombreux projets informatiques sont confiés entièrement à des équipes qui ne connaissent pas le business de l'entreprise, ce qui mènent régulièrement à des incompréhensions des deux côtés. Même dans des projets de petites tailles le rôle de Business Analyst est essentiel. C'est d'autant plus vrai lorsque le développement est confié à une équipe et se fait complètement à distance. Il est essentiel d'avoir un interlocuteur qui sache communiquer, qui connaisse le business et qui dispose de connaissances et d'outils qui permettent de traduire les besoins dans un modèle non ambigu et compréhensible par les deux parties.

Dans le cas de petits projets qui doivent être mis en production rapidement avec des requirements pouvant évoluer, le choix d'une méthode de gestion de projet classique de type waterfall n'est pas la plus pertinente. En effet, une approche Agile s'avère être plus appropriée pour un déploiement rapide. Il est important cependant de classifier dès le début ces requirements afin de dégager un « minimum viable product » qui pourra aller en production et vérifier rapidement la valeur d'affaires.

Dans le cas présenté ici l'entreprise a sous-traité son projet à une équipe basée à l'étranger. Au cours des dernières années, le recours à la sous-traitance pour des raisons de budget est en effet devenu une réalité pour les entreprises. Cependant il est nécessaire de procéder à des adaptations des méthodes Agile pour fonctionner avec ce modèle.

En effet, pour utiliser un mode de développement Agile avec des équipes délocalisés, il s'avère nécessaire de faire une pré-étude afin d'identifier le contexte, de manière à adapter la méthode Agile en conséquence. Trop souvent dans ces cas-là l'aspect culturel est mis de côté. Et pourtant c'est le premier élément à prendre en compte et à intégrer. Il est également important d'attaquer et dès le début de projet la problématique primordiale de la communication. En effet des incompréhensions peuvent avoir des impacts non négligeables sur la réussite du projet. Dans le cas des projets sous-traités à l'étranger dans lequel il y a peu de contacts physiques avec les individus il faut utiliser un langage de modélisation non ambigu, sans quoi l'efficacité de la méthodologie Agile peut être mise à mal dès le début.

Il est important également de souligner que lorsqu'on fait appel à un prestataire, on s'engage avec lui tout au long du projet, et qu'idéalement, la phase de maintenance, souvent oubliée dans des petits projets, doit se faire avec lui. C'est pourquoi il est nécessaire également de le considérer comme un partenaire de confiance et prendre le temps de bien négocier tous les aspects du contrat avec lui avant de commencer la collaboration à proprement dite.

9. Bibliographie

Andersson, M. (2015, février 24). *Start-Up Chile and the birth of 'Chilecon Valley' – changing a country's future through attraction of talent and startups*. Récupéré sur StartupChile: http://www.startupchile.org

Aubry, C. (2007, Janvier). *Le rôle de ScrumMaster*. Récupéré sur Scrum, agilité et rock'n roll: http://www.aubryconseil.com/post/2007/01/07/148-le-role-de-scrummaster

Ayed, H. (2012). *A metamodel based approach for customizing and assessing agile methods.*

Ayed, H., Habra, N., & Vanderose, B. (2013). *AM-QuICk : a measurement-based framework for agile methods customisation.*

Ayed, H., Habra, N., & Vanderose, B. (2015). *A Context-Driven Approach for Guiding Agile Adoption: The AMQuICk Framework.*

Boehm, B., & Turner, R. (2003). *Balancing Agility and Discipline - A Guide for the Perplexed.* Addison-Wesley.

Burnay, C. (2016). *IBAGM321 - Ingénierie des exigences.*

Davinson, D. (2003). Top 10 Risks of Offshore Outsourcing.

Dejean, K. (2016). *IBAGM312 - Organisation et gestion du changement.*

DeYoe, P. (2009). *Can Agile be Combined with Offshore? 12 Lessons.*

Englebert, V. (2017). *IBAGM322 - Modélisation organisationnelle et métier : langages et méthodes.*

Fowler, M. (2006). *Using an Agile Software Process with Offshore Development.*

Hofstede, G. (s.d.). *Culture compass - Country comparison*. Récupéré sur https://geert-hofstede.com/countries.html

Hofstede, G., Hofstede, G. J., & Minkov, M. (2010). *Cultures and Organizations, Software of the Mind. Third Edition.* London: McGraw-Hill.

Jeff Sutherland, G. S. (2008). *Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams .*

Jeff Sutherland, K. S. (2001). *The Agile manifesto.*

KAMDEM, G. D. (2016, septembre). *Qu'est-ce que la méthode agile apporte à une organisation ?* Récupéré sur https://fr.linkedin.com/pulse/quest-ce-que-la-m%C3%A9thode-agile-apporte-%C3%A0-une-ga%C3%ABlle-diane-kenmogne

Karpman, S. (1968). Fairy tales and script drama analysis.

Kolp, M. (2016). *IBAGM323 - Gestion de projet et gestion de risques.*

Krivitsky, A. (2011). Offshore Outsourcing with Scrum. *AGILEEE 2011.*

Kruchten, P. (2000). *The Rational Unified Process (RUP): An Introduction (second edition).* Addison-Wesley.

Kruchten, P. (2010). *Contextualizing Agile Software Development.* Vancouver.

Maquet, B. (2015). *IBAGM313 - Négociation et analyses des conflits.*

Messager Rota, V. (2008). *Gestion de projet vers les méthodes agiles.* Eyrolles.

Moore, S., & Barnett, L. (2004). *Offshore Outsourcing And Agile.*

O'Neill, E. (2005). *Conduite de projets informatiques offshore.* Eyrolles.

Pichler, R. (2010, Mars 9). *Business Analyst in Scrum*. Récupéré sur Scrum Alliance: https://www.scrumalliance.org

Pichler, R. (2010, Avril 2010). *Common Product Owner Traps*. Récupéré sur Scrum Alliance: https://www.scrumalliance.org

Roques, P. (2009). *UML 2 par la pratique.* Eyrolles.

Sieberath, R. (2015). *IBAGM300 - Business model design et innovation - Projet d'analyse.*

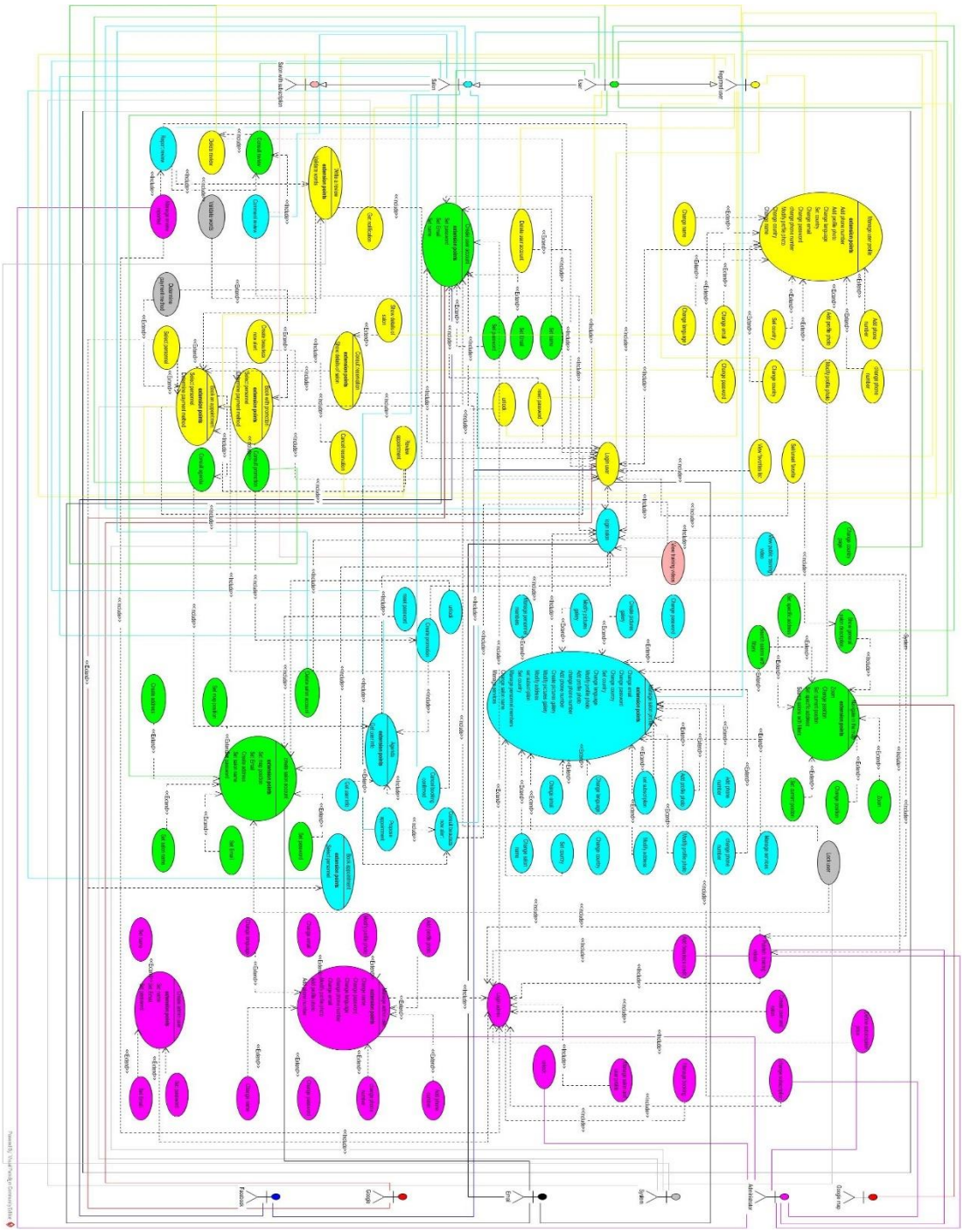
Thiran, P. (2017). *IBAGM311 - Théories et Stratégies d'innovation en ICT.*

Van Enis, Q. (2017). *IBAGM333 - Droit des TIC.*

Wautelet, Y. (2016). *IBAGM331 - Stratégies IT et qualité des services.*

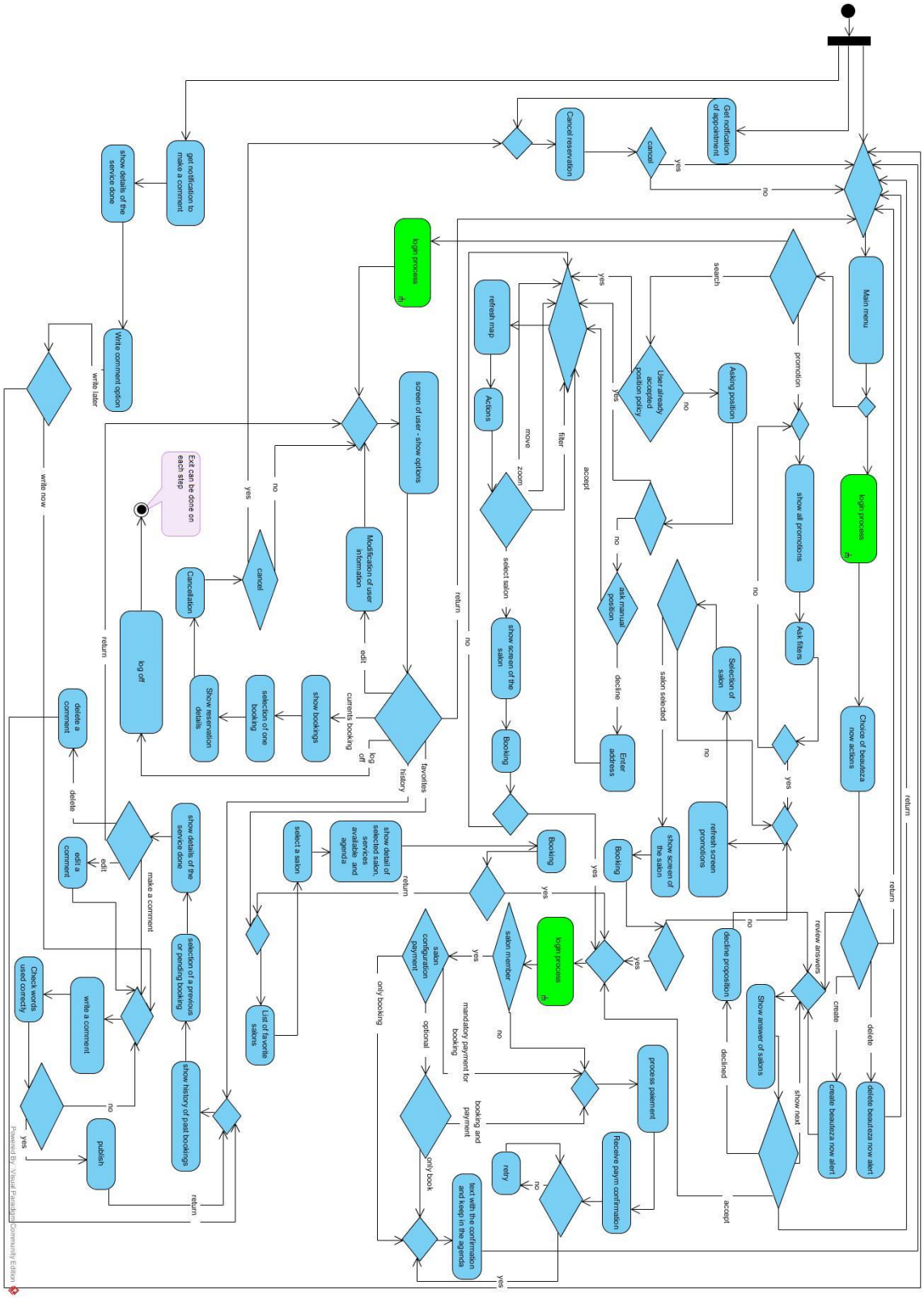
10. Annexes

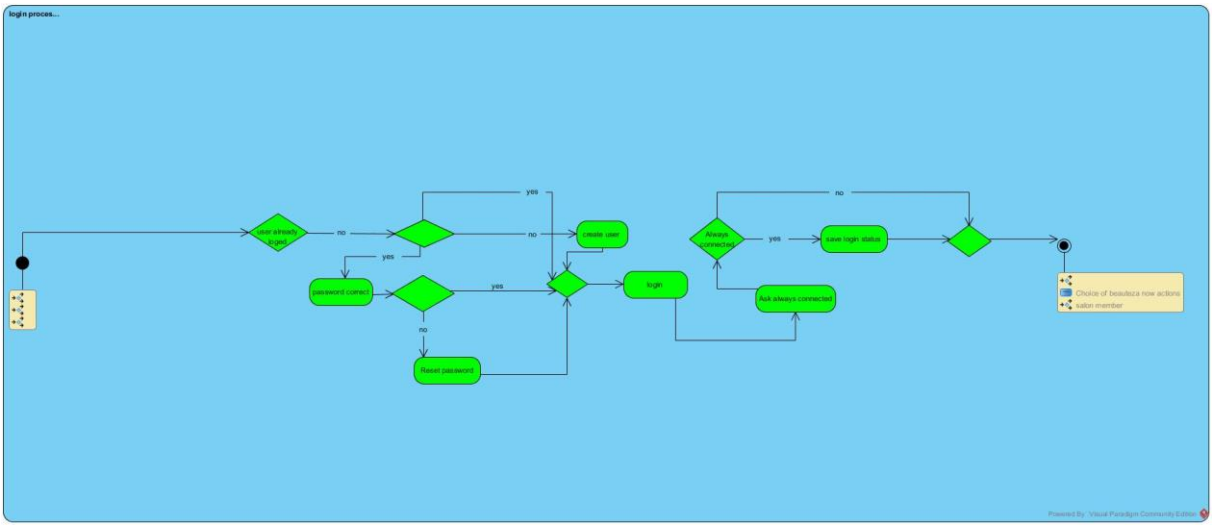
10.1. Use case diagram



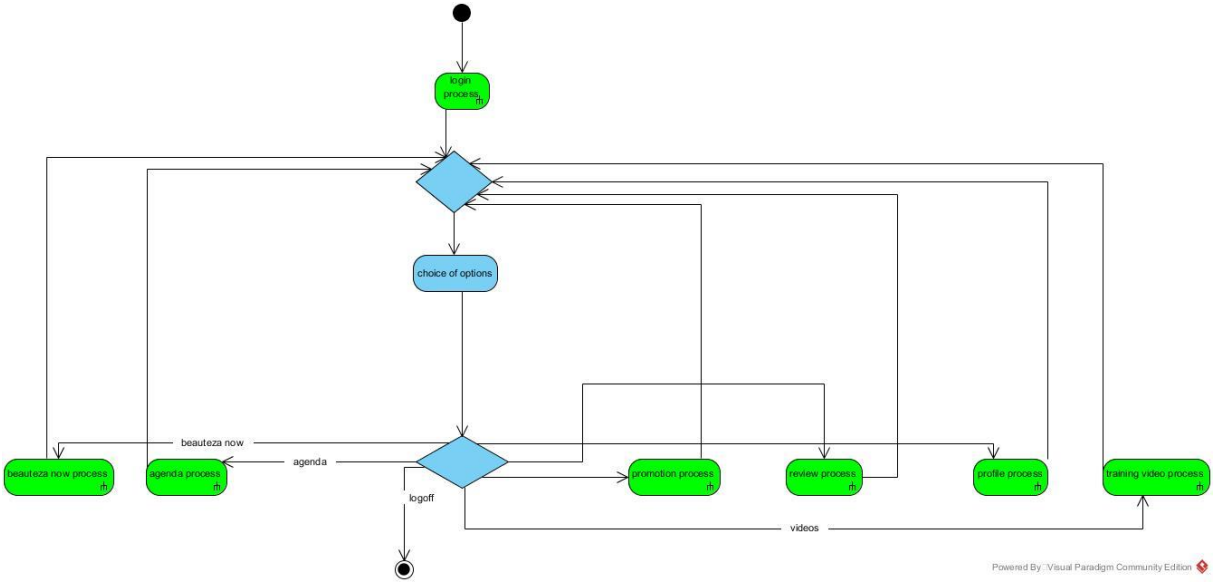
10.2. Activity diagram

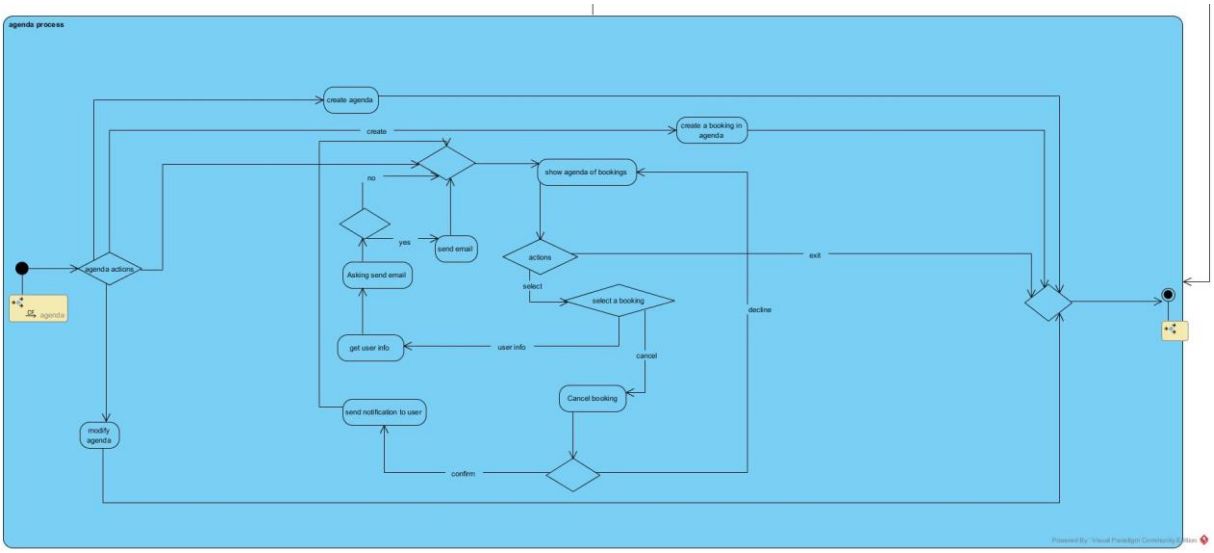
10.2.1. User

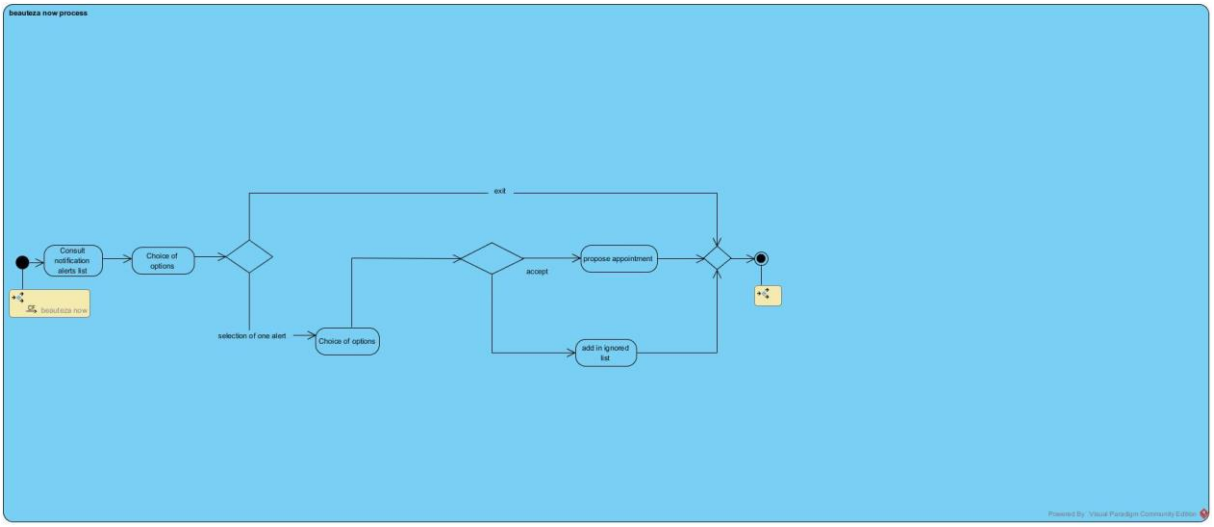


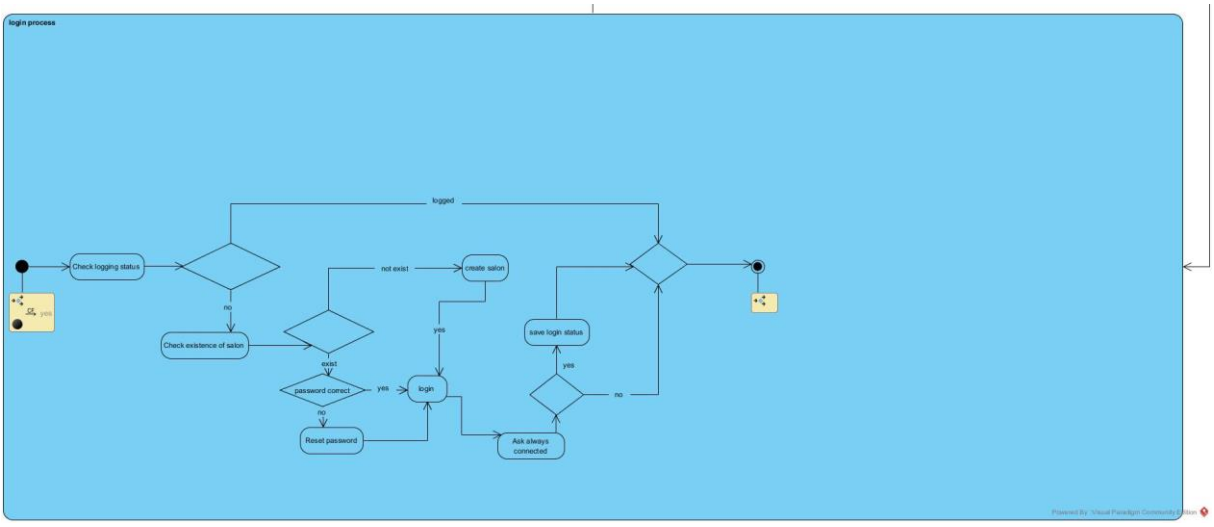

login proces...
user already logged
no
yes
no
yes
password correct
yes
no
Reset password
create user
login
Always connected
no
yes
save login status
Ask always connected
Choice of beauteza now actions
salon member

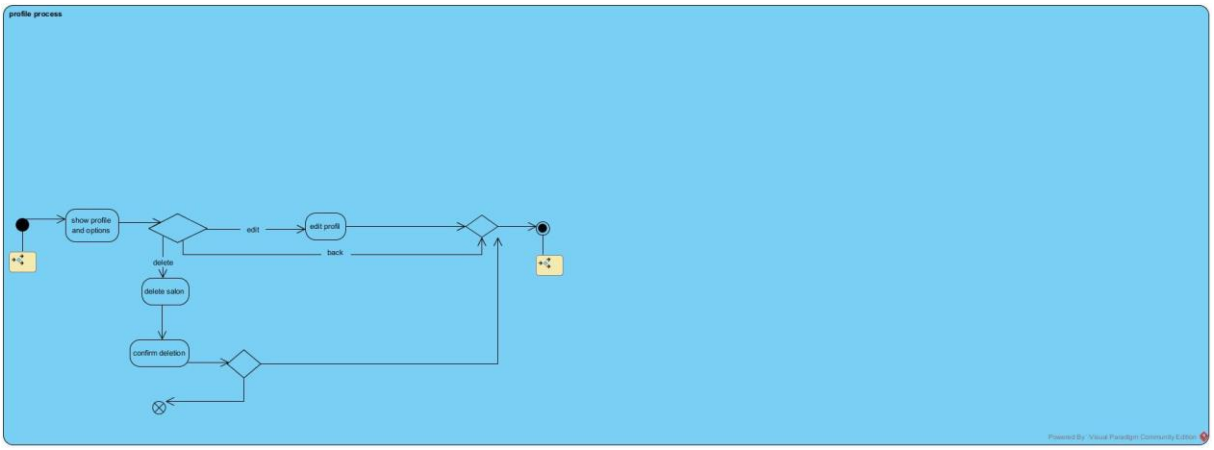
10.2.2. Salon

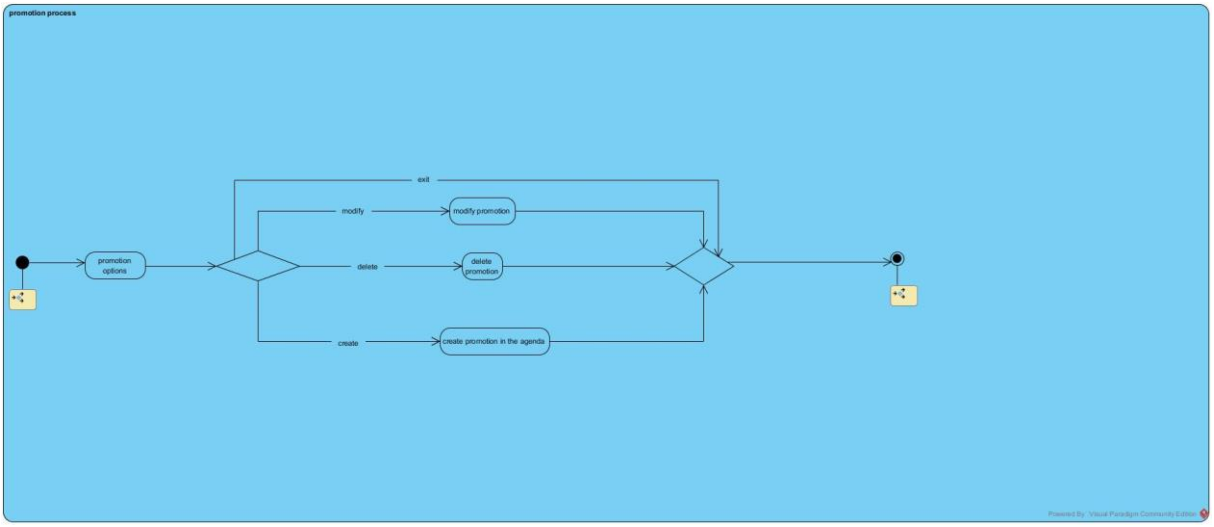

login process
choice of options
beauteza now
agenda
logoff
videos
beauteza now process
agenda process
promotion process
review process
profile process
training video process
Powered By Visual Paradigm Community Edition

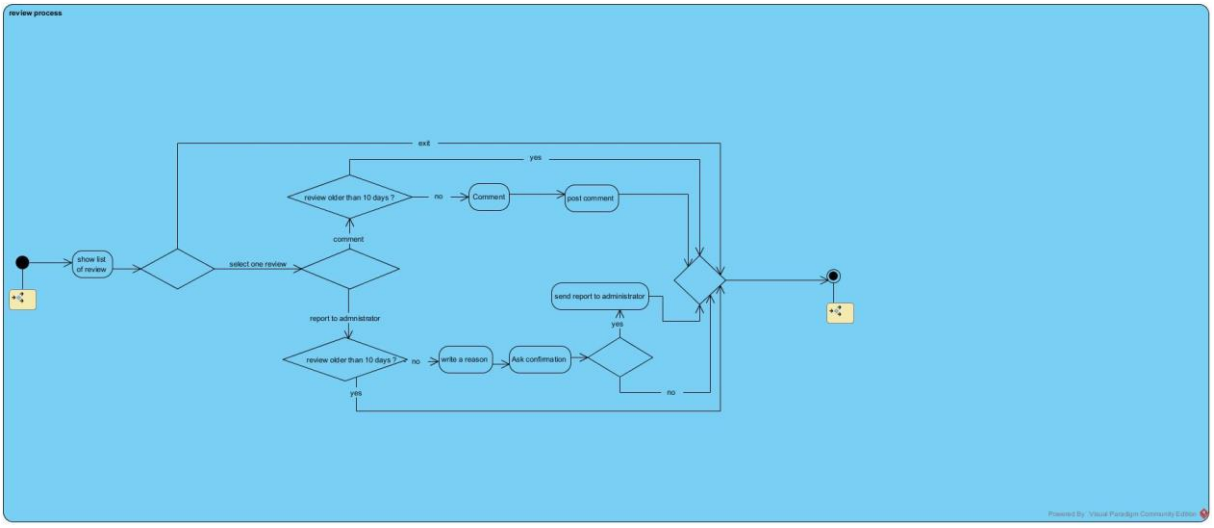

agenda process
create agenda
create
create a booking in agenda
show agenda of bookings
no
yes
send email
Asking send email
agenda actions
actions
exit
select
select a booking
decline
get user info
user info
cancel
Cancel booking
send notification to user
modify agenda
confirm
agenda

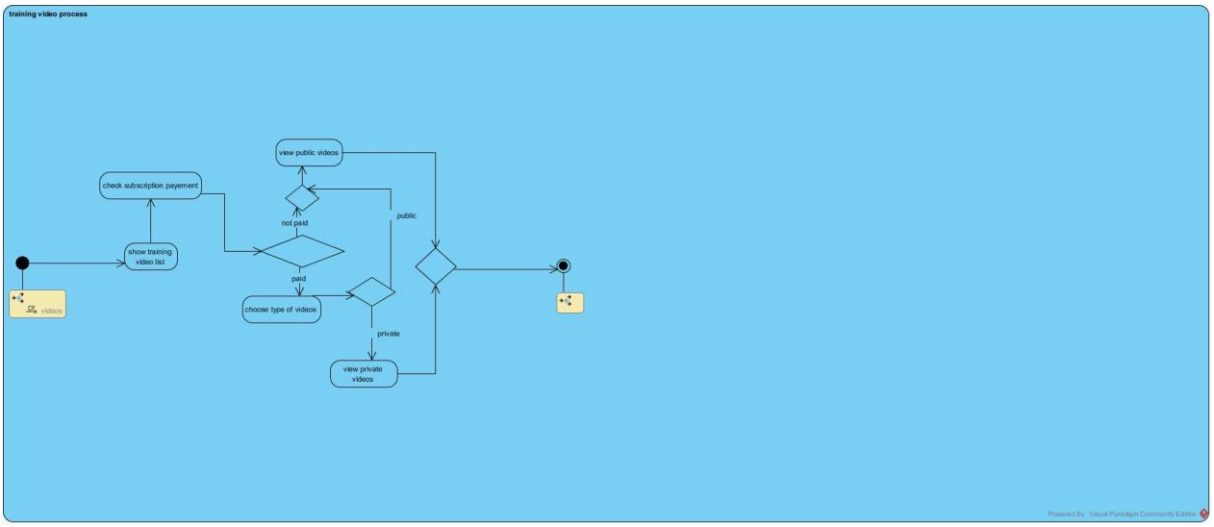

beauteza now process
exit
Consult notification alerts list
Choice of options
accept
propose appointment
selection of one alert
Choice of options
add in ignored list
beauteza now


login process
logged
Check logging status
not exist
create salon
no
Check existence of salon
yes
save login status
exist
password correct
yes
login
yes
no
no
Reset password
Ask always connected
yes

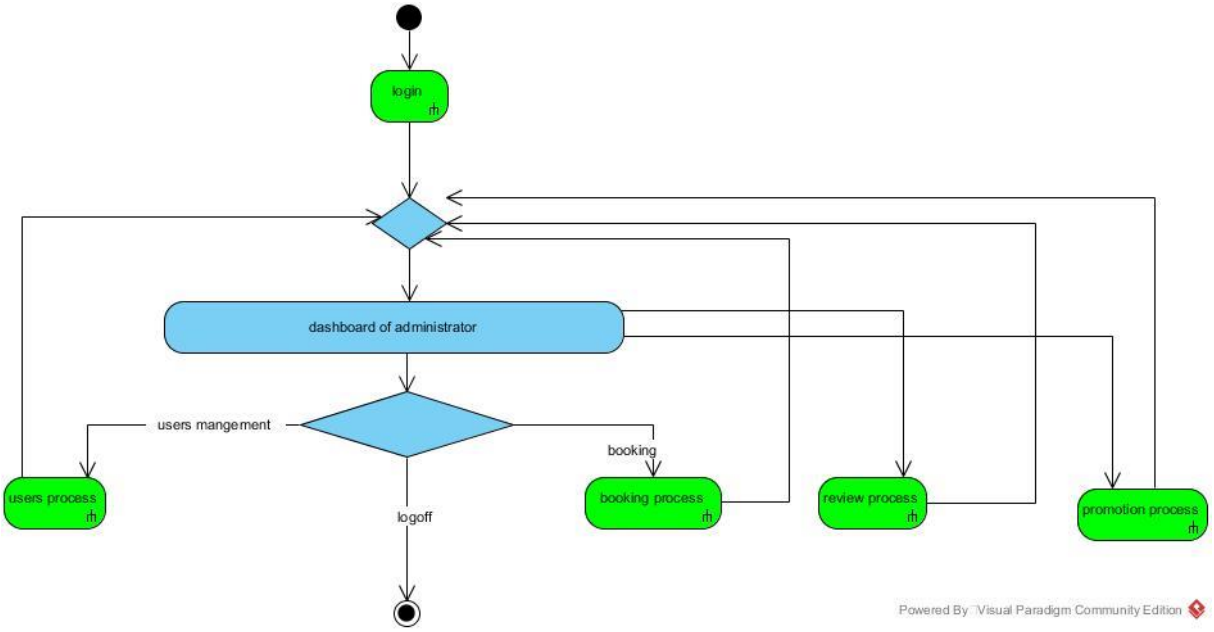

profile process
show profile and options
edit
edit profil
back
delete
delete salon
confirm deletion

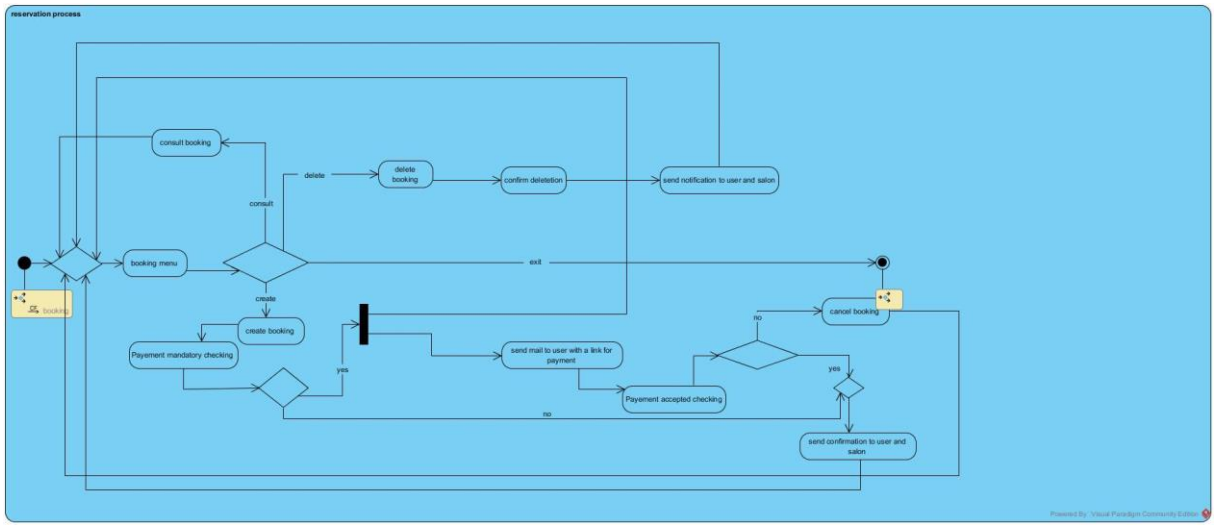

promotion process
promotion options
exit
modify
modify promotion
delete
delete promotion
create
create promotion in the agenda

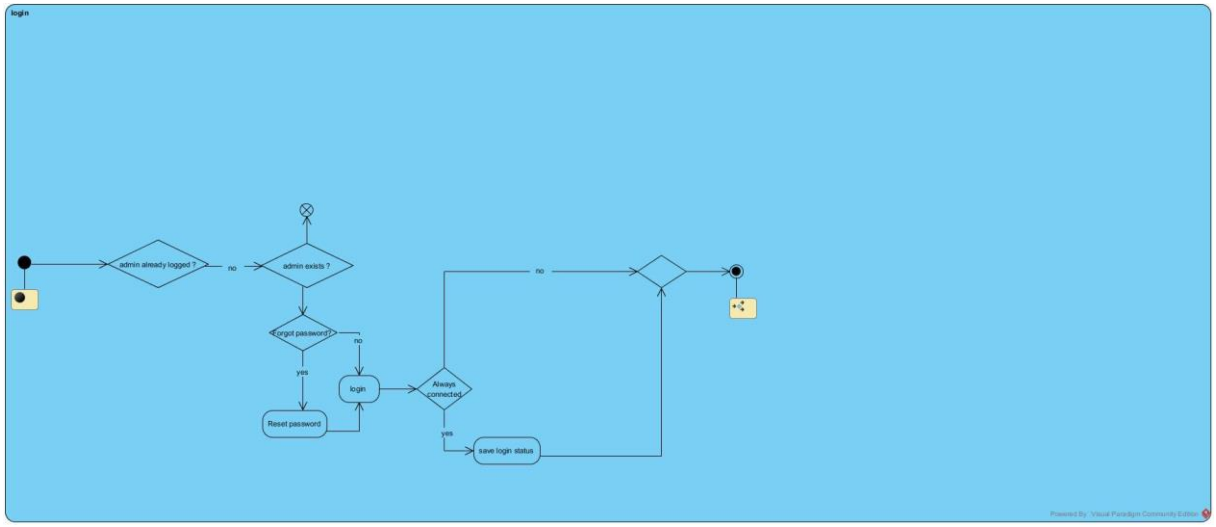

review process
exit
yes
review older than 10 days ?
no
Comment
post comment
comment
show list of review
select one review
send report to administrator
report to administrator
yes
review older than 10 days ?
no
write a reason
Ask confirmation
yes
no

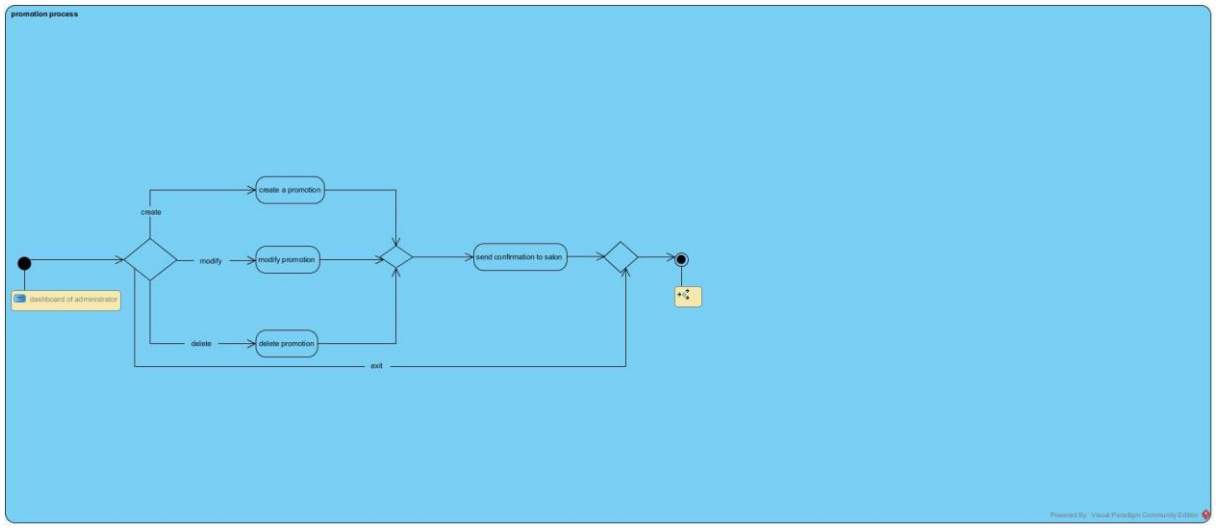


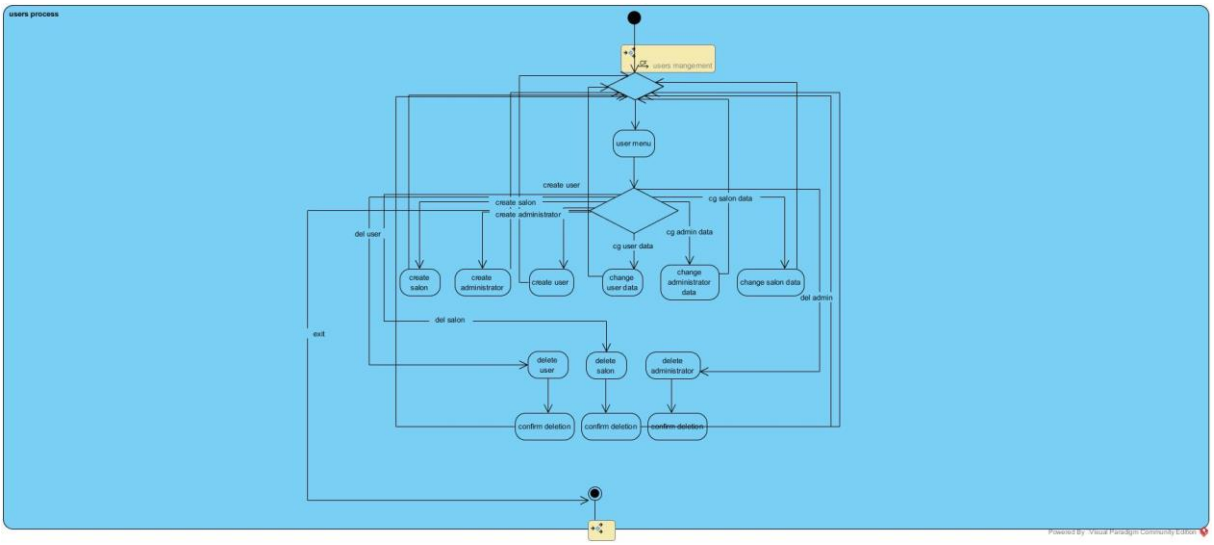
10.2.3. Administrator



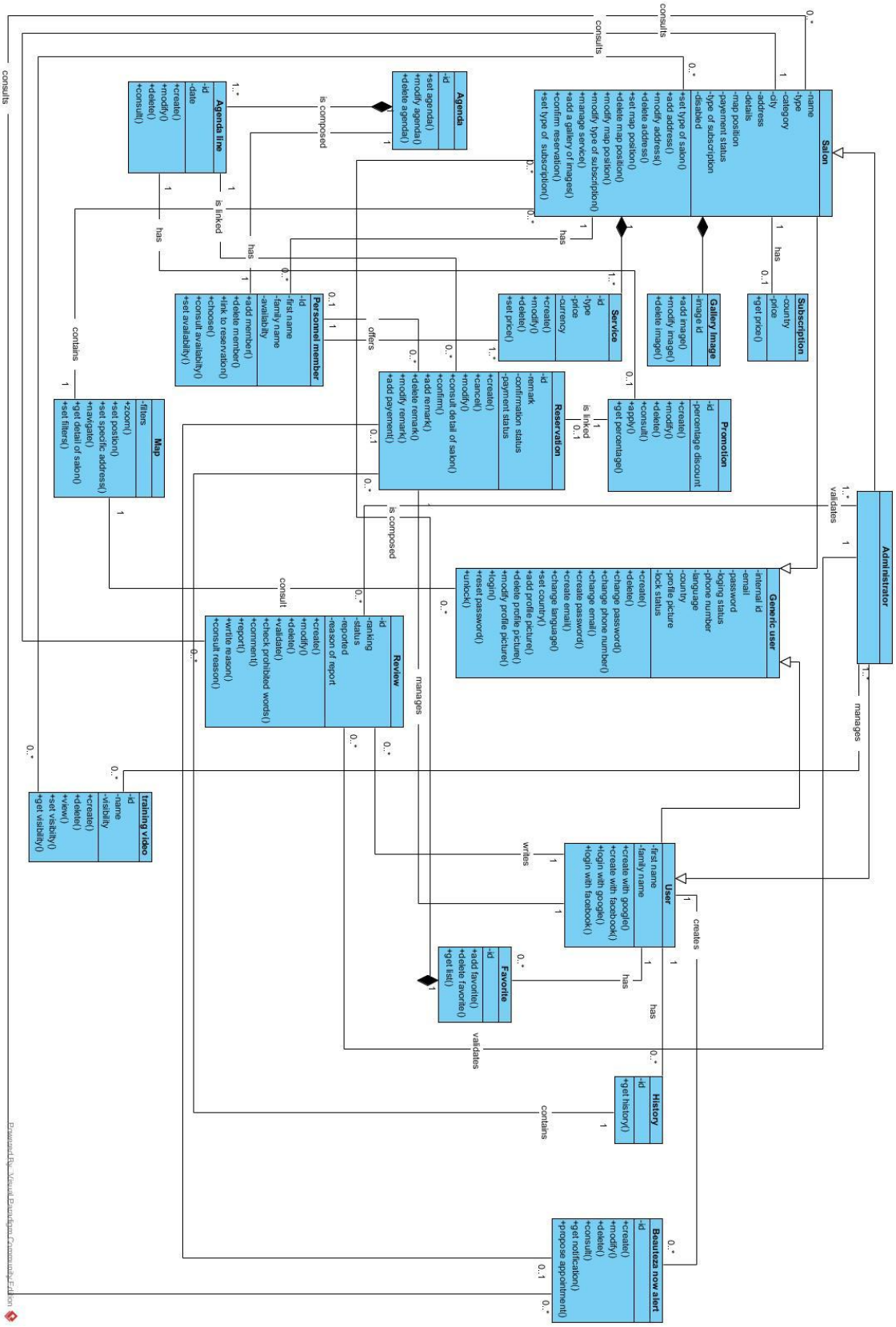

reservation process
consult booking
delete
delete booking
confirm deletetion
send notification to user and salon
consult
booking menu
exit
booking
create
create booking
Payement mandatory checking
yes
send mail to user with a link for payment
no
cancel booking
yes
Payement accepted checking
no
send confirmation to user and salon


login
admin already logged ?
no
admin exists ?
Forgot password ?
no
yes
login
Always connected
no
yes
Reset password
save login status


promotion process
create a promotion
create
modify
modify promotion
send confirmation to salon
dashboard of administrator
delete
delete promotion
exit

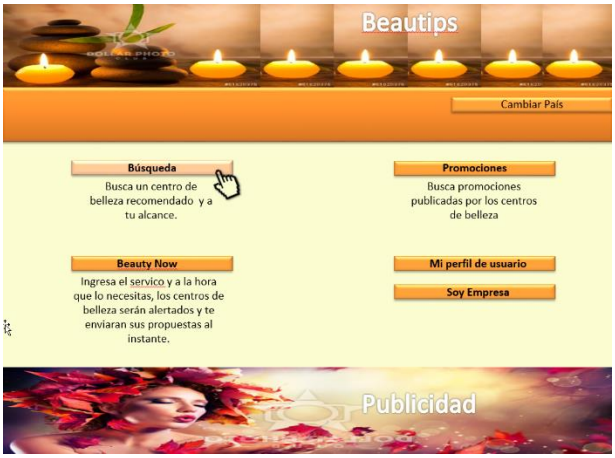

users process
users mangement
user menu
create user
create salon
create administrator
cg salon data
cg admin data
cg user data
del user
create salon
create administrator
create user
change user data
change administrator data
change salon data
del admin
del salon
exit
delete user
delete salon
delete administrator
confirm deletion
confirm deletion
confirm deletion

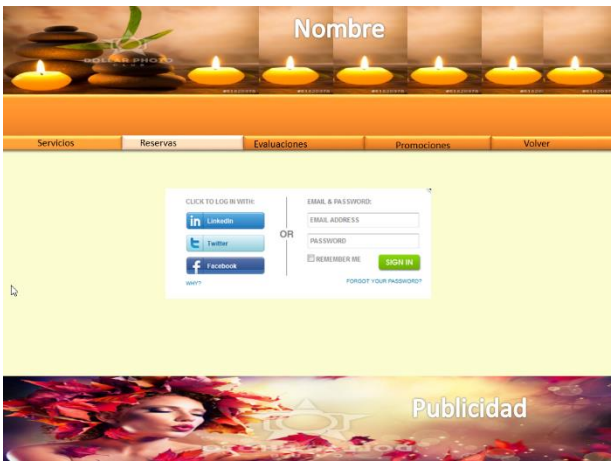
10.3. Class diagram



10.4. Mockup

Voici quelques extraits du mockup réalisé :





10.5. Test cases

Extrait d'un tests case réalisé :

Test Case#	1	User creation				Status		
Description								
	Create a user, login, delete user, login					NOK		
Pre-Requisites								

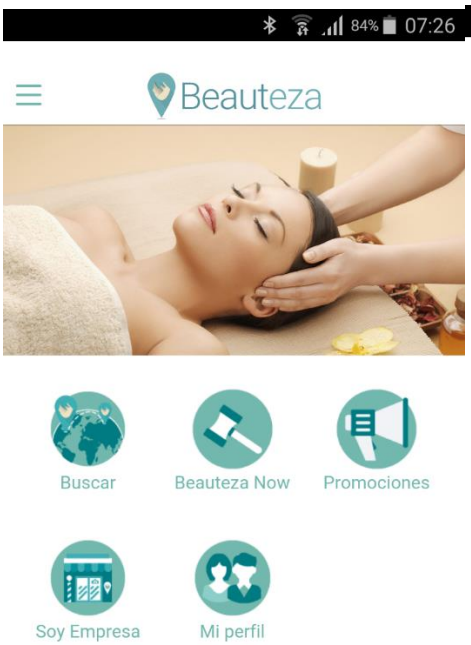
Execution Date	Executed By	Release
14/11/2016	Olivier	2

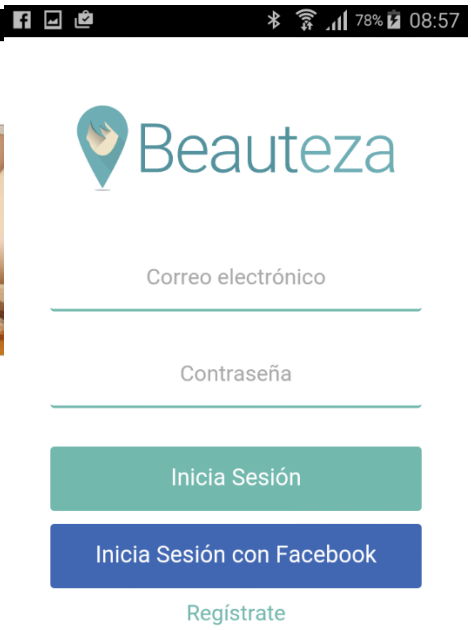
Use case involved
create user account
Delete user account
Login user

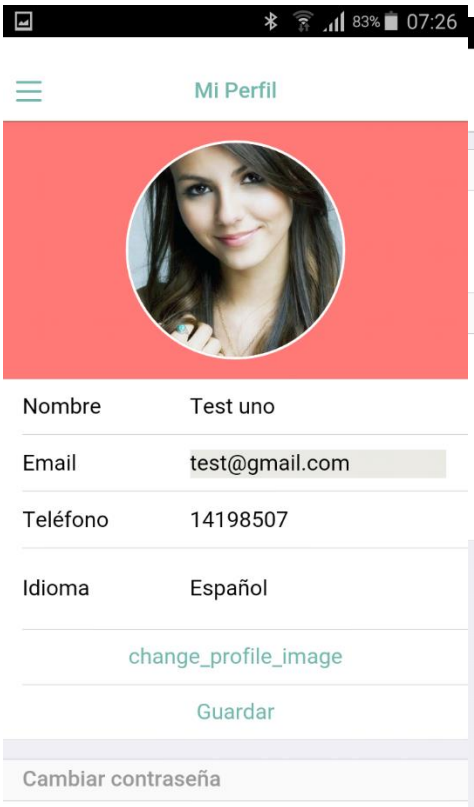
Test Script#	1							
Step#	Action	precondition	xpected Resul	Test Data	Actual Result	Result	Defect(s)	Comments
1	create a user		user created		a phone number is requested, tried to set blank first and there is a check on this number, tried to enter 0047898272 7(correct BE number with intl	KO		
1.1	set Name					ok		
1.2	set					ok		
1.3	set email					ok		
2	login	user	logged			KO		

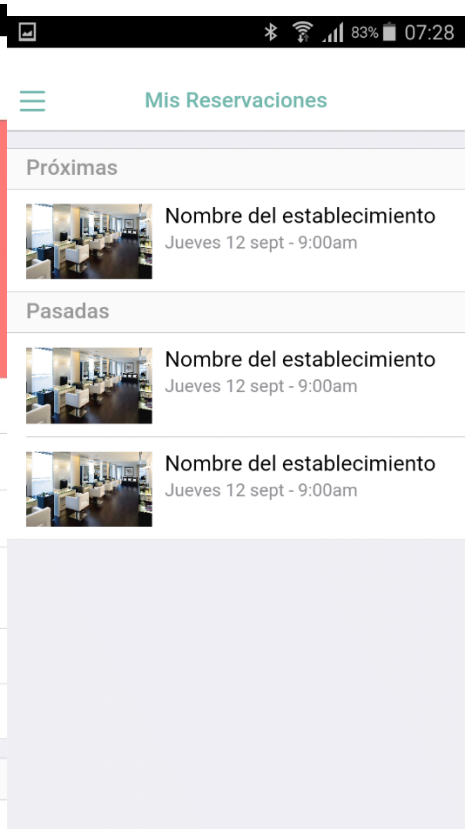
Cover | Test case 1 | Test case 2 | Test case 3 | Test case 4 | Feuil4

10.6. Captures d'écran de la première release









10.7. Matrice MoSCoW

MoSCoW Beauteza

REQUIREMENTS	MUST (M)	SHOULD (S)	COULD (C)
Website user			x
IOS application user	x		
Android application user	x		
Windows phone application user			
Website salon	x		
IOS application salon		x	
Android application salon		x	
Windows phone application salon			
Map navigation		x	
Register Salon	x		
Register user	x		
Register admin			x
admin account	x		
Delete user account	x		
Delete salon account	x		
Delete admin account	x		
login user	x		
Login user facebook		x	
Login user google		x	
login salon	x		
Login salon facebook			x
Login salon google			x
login admin	x		
Salon without subscription			x
Salon with Subscription	x		
Manage Salon profile	x		
Manage user profile	x		
Manage admin			x
Beauteza now	x		
Interraction with google/apple agenda for user			x
Agenda salon			x
Planning employees salon			x
Appointement management		x	
Choose employee with appointement			x
Favorites			x
promotions salon			x
electronic payement			x
review management			x
admin management	x		
training videos			x
English support	x		
Spanish support	x		
French Support			x
Portugese support			x
Chinese support			